RL-TR-95-171
Final Technical Report
September 1995

# RESEARCH DIRECTIONS IN DATABASE SECURITY VI: PROCEEDINGS OF THE SIXTH ROME LABORATORY MULTILEVEL DATABASE SECURITY WORKSHOP - 22-24 JUNE 1994

SRI International

LouAnna Notargiacomo (Editor)

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

RL-TR-95-171 has been reviewed and is approved for publication.

APPROVED: [illegible]

JOSEPH V. GIORDANO
Project Engineer

FOR THE COMMANDER: [illegible]

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

| | | |
|---|---|---|
| 1. AGENCY USE ONLY (Leave Blank) | 2. REPORT DATE<br>September 1995 | 3. REPORT TYPE AND DATES COVERED<br>Final ---- |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>RESEARCH DIRECTIONS IN DATABASE SECURITY VI: PROCEEDINGS OF THE SIXTH ROME LABORATORY MULTILEVEL DATABASE SECURITY WORKSHOP - 22-24 JUNE 1994 | 5. FUNDING NUMBERS<br>C - F30602-94-C-0238<br>PE - 33401G<br>PR - 1068<br>TA - 01<br>WU - P9 |
| 6. AUTHOR(S)<br>LouAnna Notargiacomo (Editor) | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>SRI International<br>333 Ravenswood Avenue<br>Menlo Park CA 94025-3493 | 8. PERFORMING ORGANIZATION REPORT NUMBER<br>N/A |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>Rome Laboratory (C3AB)<br>525 Brooks Rd<br>Griffiss AFB NY 13441-4505 | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER<br>RL-TR-95-171 |

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Joseph V. Giordano/C3AB/(315) 330-3681

| | |
|---|---|
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution unlimited. | 12b. DISTRIBUTION CODE |

13. ABSTRACT (Maximum 200 words)

This report documents the proceedings of the Sixth Workshop on Database Security, which was sponsored by Rome Laboratoy. This workshop was held in Southwest Harbor, Maine, during 22-24 June 1994, and is the sixth in a series of database security workshops sponsored by Rome Laboratory. This workshop addressed current results of research projects in the area of multilevel database security. The topics addressed included multilevel database models, object-oriented database management, multilevel secure database management system architectures, system assurance, distributed and federated secure database management, inference control, usability of trusted database management systems, and the transition of technology from the research community to products and systems.

| | |
|---|---|
| 14. SUBJECT TERMS<br>Multilevel database security | 15. NUMBER OF PAGES<br>184 |
| | 16. PRICE CODE |

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>UL |

## ACKNOWLEDGMENTS

This workshop was sponsored by Rome Laboratory under sponsorship from Joseph V. Giordano. We would like to thank Mr. Giordano for his support and participation in this workshop. We would also like to thank Teresa Lunt for her involvement in organizing and co-chairing this workshop. Finally, we would like to thank the participants of this workshop for their papers, presentations, and discussions.

About the Authors and Discussion Leaders:

Rae Burns,[a] AGCS, Inc.
Oliver Costich, Naval Research Laboratory
Judith Froscher, Naval Research Laboratory
Tom Garvey, SRI International
Thomas Haigh, Secure Computing Corporation
William R. Herndon, The MITRE Corporation
T. Y. Lin, San Jose State University
Teresa Lunt,[b] SRI International
Donald Marks, National Security Agency
Catherine D. McCollum, The MITRE Corporation
LouAnna Notargiacomo, The MITRE Corporation
Dick O'Brien, Secure Computing Corporation
James P. O'Connor, Infosystems Technology, Inc.
Xiaolei Qian, SRI International
Ravi Sandhu, George Mason University
Marvin Schaefer, ARCA Systems, Inc.
Kenneth P. Smith, The MITRE Corporation
Dan Thomsen, Secure Computing Corporation
Sandra Wade, ONTOS
Simon Wiseman, U.K. Defense Research Agency
Jack Wool, ARCA Systems

| Accesion For | | |
|---|---|---|
| NTIS CRA&I | ☒ | |
| DTIC TAB | ☐ | |
| Unannounced | ☐ | |
| Justification | | |
| By | | |
| Distribution / | | |
| Availability Codes | | |
| Dist | Avail and / or Special | |
| A-1 | | |

[a] Rae Burns is currently employed with AGCS, Inc. At the time of the workshop, she was employed with GTE Government Systems.

[b] Teresa Lunt is currently at the Advanced Research Projects Agency (ARPA).

TABLE OF CONTENTS

TITLE PAGE

## MESSAGE FROM THE WORKSHOP CHAIRS

### OVERVIEW OF THE WORKSHOP

The 1994 Rome Laboratory Workshop, Research Directions in Database Security VI, was held from 22-24 June 1994 at the Claremont Hotel in Southwest Harbor, Maine. This workshop is the sixth in a series sponsored by Rome Laboratory.

The focus of this workshop was to both look at the current state of research advances in various areas of database security, but to also look at the needs of potential users of this technology and ways to transition research results to products or fielded systems. This workshop consisted of the presentation of research results in various database technology areas and corresponding discussions assessing the state of technology and the requirements for further investigation. Presentations and discussions were held on the following topics:

- Multilevel Secure Database Models
- Object-Oriented Database Management
- Multilevel secure Database Management System Architectures
- Assurance
- Distributed/Federated Secure Database Management Systems
- Technology Transition
- Inference Control
- Usability of Multilevel Secure Database Management Systems

LouAnna Notargiacomo
Workshop Co-Chair

Teresa Lunt
Workshop Co-Chair

## MULTILEVEL SECURE DATABASE MODELS

# SEMANTICS OF SECURITY CLASSIFICATIONS
## Critical Review of MLS Data Model

T. Y. Lin

Mathematics and Computer Science
San Jose State University
San Jose, CA 95192

## 1. INTRODUCTION

There are many misconception about the meaning of a security label and MLS data model. We will discuss the meaning and structure of security labels. A MLS relational databases is a mathematical model of a secure world. From that we derive the meaning of security label:

**Labeling Principle: The security label of a data is the security label of the corresponding fact of the real world.**

In particular, the label of an element is the label of a property of an entity. Moreover, in a secure world, every objects or entities are classified, so we conclude from the meaning of mathematical modeling that all information in a MLS database should be labeled:

**Data Labeling Principle: All views and their instances should have their own security labels.**

If a MLS relational database is a Bell-LaPadula model, then we conclude that

**Bell-LaPadula Labeling Principles: Bell-LaPadula Model implies that all views and their instances should be labeled.**

So we have the same conclusion as what should be labeled from two different sources. A relational database is represented by a set of base relations. From the literature on database theory we conclude that

**Equivalency Principle: Two sets of base relations define the same relational database iff they define the same set of views**

A data in a mathematical model corresponds to a unique object in the real world.

**Labeling Integrity Principle: A data can only be receive on label.**

Since the foreign key in guest table and the corresponding primary key at home table represent the same fact in real world, in fact, they are represented by the same element in the universal relation, so they should receive the same label:

Referential Integrity Labeling Principle: The label of a foreign key (in guest table) should equal to the label of the corresponding primary key at home table.

Finally, we point out some weak points in the "standard" element-level-labeling or tuple-level-labeling MLS data model.

## 2. SEMANTICS OF SECURITY LABELS

### 2.1 What is a data?

A data is a mathematical representation of a piece of the real world. Each data represents certain portion of the real world: A primitive data represents a primitive fact, a complex data represent a complex fact. So the fundamental principle of the data labeling is :

Labeling Principle: The security label of a data is the security label of the corresponding fact of the real world.

The principle sounds obvious,(it was mentioned many times in SeaView documents [Lunt89]) in practice, there are misconceptions.

Digression.
Here we are using the notion of data modeling. "In data modeling we try to organize data so that they represent as closely as possible the real world situation" [TsLo82, pp.1]. On the other hand, in commercial world, data model often refers to the collection of the data types (templates) of all the base relations. So data model of commercial world seems have no data but have data structure only. The two notion sounds different. What is data "type"? In entity relationship model, one often define the entity "type" as the set of entities (under discussion) with the same attributes [Emas94, p.46]. Similarly, the data type, mathematically, should be understood as the set of the data (under discussion) with same structure, even though it is often interpreted as a template (or data structure) in a computer memory in commercial world. Certainly, the template (in computer memory) is intend to hold the data under discussion. So one can think of template as the set of intended data. Therefore, intrinsically both notions agree. To avoid confusing, we will use data models only in the sense of [TsLo82] through out the whole paper.

### 2.2. Elements vs Raw Data.

In the relational model, the element is the most primitive data. Because of its primitiveness, its real world meaning is often miss-interpreted. Though an element is a data from the domain, it is not a raw data. In the following relation, the **first "50,000"** represents one aspect of the entity Mr. Smith. So if we do label the element 50,000 we are **not** labeling 50,000 per se, it is Mr. Smith's 50,000. In general a tuple represents an entity in the real world, and an element represents a property of the entity. So the label of an element is a label of that property in the real world. A raw data (an element in the domain) can not be assigned a label, because it does not have a real world meaning.

Example 1.

RECORD

| Name | CL1 | Salary | CL2 | Telephone | CL3 | Occupation | CL4 |
|---|---|---|---|---|---|---|---|
| Smith | S | 50,000 | S | 123-456-7890 | S | Physicist | S |
| Peterson | S | 65,000 | S | 321-654-0987 | S | Nuclear Engineer | S |
| Jones | U | 50,000 | U | 123-654-0987 | U | Accounting | U |

There are two labels for 50,000 in this relation. This does not mean that the security labeling is inconsistent or multi-labeling. It merely means that the raw data 50,000 was used twice, first is representing Smith's salary, second Jone's salary. Element labeling is never meant to be the labeling of raw data. In database, a tuple instance often represents some real world entity and its elements represent attributes (properties) of the entity. *The security label of an element represents the labeling of this real world meaning (a property of the entity).*

## 2.3 Labeling of Complex Data

Let us consider the following query

```
SELECT Name, Salary
FROM    Record
WHERE   Name=Smith
```

The output data consists of "Smith" and "50,000". How these two data pass through the reference monitor (the computer module which checks the security labels). Is it check

Case (1): one element at a time or
Case (2): two data together?

In Case (1), it checks two simple facts, namely the two elements, while in Case (2), it check one complex fact, namely the tuple (Smith, 50,000).

Which is more meaningful? We believe Case (2) is more meaningful. Mathematically,

Case (1) implies that the system has no notion of aggregate (no set theory)
Case (2) has aggregate, therefore a complex fact should receive a new label.

By pushing the Case (2) to its full strength, we believe we have

**Data Labeling Principle: All views and their instances should have their own security labels.**

## 2.4. Computational Complexity

If an instance of a database has n elements, then potentially (in worst case) there are $2^n$ view instances. So labeling is an exponential problem. A complete labeling to all the view instances is a formidable task. Some automation is needed or mathematically, it needs a structure. Fortunately, Denning did introduce the notion of lattice model [Denn76] which we have extended it to aggregated security algebra. The mathematical structure of these labels or security classes allows us to label view instance "on-the-fly" [Lin90b].

## 3. THE IMPLICATION OF BELL-LAPADULA MODEL
-- The Data Model As A Bell LaPadula Model.

What Should be Classified? In Bell-LaPadula Model (BLM), every object or subject is assigned a security class(label). Now if we apply BLM to database systems, then BLM requires that every object processed by database systems should have security label. What are the objects processed by databases?

(1) Intentional Objects: They are objects in Data Dictionary, such as, names of attributes, relation schema, query statements, and constraints. Security classes of intentional objects hide the existence of high data.

(2) Extensional Objects: They are the elements, tuples, relations, view instances, and relational algebraic expressions.

So a MLS data model is a special case of Bell-LaPadual, therefore all view instances should be labeled. So we have reached the same conclusion as in Section 2

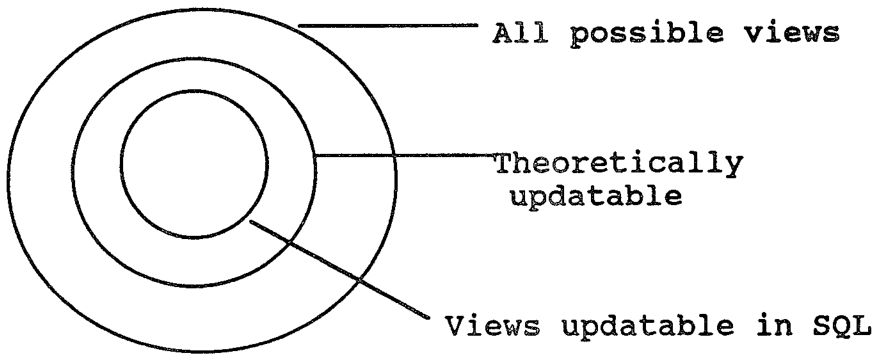
**Bell-LaPadula Labeling Principles: Bell-LaPadula Model implies that all views and their instances should be labeled.**

## 4. VIEWS AND A RELATIONAL DATABASE

There are misconceptions about relational data base.

Myth: A relational data base is a set of base relations

A relational data base is represented by a set of base relations, but such a representation is not unique. In other words, the same data base may be represented by another set of base relations. When two sets of base relations represent the same database? The answer is if they give the same set of views.



Although there is no official answer in the literature, however, the answer is implicit in the literature. The so called normalization is a process of decomposing a universal relation into a set of the "best" base-relations-representation, such as Boyce Codd Normal Form (BCNF) or 5th Normal Form (5NF). In the process, a series of new sets of base relations is constructed. Database designers believe that the series of base relations although are different, they define the same relational database. Fundamentally, what can a user sees from a given database ? The set of all possible views. So if the set of all possible views is the same, from users' point of view, they are the same.

**Equivalency Principle: Two sets of base relations define the same relational database iff they define the same set of views.**

## 5. REFERENTIAL INTEGRITY

The referential integrity problem is important in security e.g., [Burns90]. We will examine it through the mathematical modeling. As mentioned above, the normalization process decomposes a universal relation into a set of "best" base-relations-representation. Such decomposition certainly introduces replica of a data into tables. Foreign keys are such examples. A foreign key is a primary key of, say, a home table H appears in a "guest" table G. A foreign key value in both tables H and G were the same element in the original table before decomposition and represents the same real world fact, so, by our principle, they should receive the same security label.

**Referential Integrity Labeling Principle: The label of a foreign key (in guest table) should equal to the label of the corresponding primary key at home table**

## 6. LABELING INTEGRITY

### 6.1. Tolerance of Inconsistency

The logical system adopted by natural science and engineering has very no tolerance on inconsistency. Let us consider the following sentence

S1: If (x is not equal to x), then (any conclusion is true).

In the traditional logic system, this sentence is a valid statement. However, the conclusion is not necessary a true statement until one can established the condition is a true statement (modus ponens). In mathematical modeling, the underlying hypotheses is that the model axioms are true statements, and its general goal is to infer more true statements. If there is any inconsistency in the axioms of mathematical model, we can never be sure that any conclusion is valid. The choice of such a logical system is, of course, a philosophical issue; we could adopt other systems. However, if we do decide not to use the traditional logic, then we have to redevelop "mathematics" and "science" based on the new logical system(at least the portion that are used in our secure system). Obviously this is not feasible. So we should stick to the traditional logical system.

A data in a mathematical model corresponds to a unique object. The data or the symbol can be regarded a name of that object.

Labeling Integrity Principle: **A data can only be referred to a unique object.**

The number 50,000 in Example 1 represent two data, one is Smith' 50,000, the other is Jones'. If the domain is implemented in the database, then there is another data 50,000 in the domain of Salary. Although, these data are all denoted by 50,000, they are all different data mathematically. Their complete description is different, only their "short hand" looks the same. Moreover, this principle is not in conflict of multi-valued modeling. There, a data refers to a unique set of objects.

### 6.2 Multi-Labeling

Some authors believe a data can be given two labels. This is dangerous, it implies that the data $X \neq X$ (two labels means a data has two different meanings). As discussed above, in traditional logic, such situation is not tolerated. In fact, mathematician uses that inequality to represent the empty set; the empty set = $\{X: X \neq X\}$. The multi-labeling is invented erroneously for dealing with composite labeling and context labeling.

### 6.1.2. Composite Labeling.

"Some thing about James Bond is top-secret, and some other thing is unclassified. So James Bond should have two security labels" said by some colleague. Let us translate this problem into much more precise form. Suppose we want to model the human being James

Bond in a computer system. So we create a document in which a paragraph, say A-paragraph, is classified top-secret and a paragraph, say C-paragraph, is unclassified. Should we assign the document two labels, simply because there are top-secret portion as well as unclassified portion ?. We say no. Our classification scheme will give the A-paragraph top secret, C-paragraph unclassified, and the total document - the "composite" document- the least upper bound of A and C paragraphs. Incidentally, this implies that the structure of security labels is important (see [Lin89c, 90b] for security algebra). Mapping back to the original problem, the some thing about James Bound should be labeled top-secret, and some other should be unclassified. The total thing about James Bond should be labeled top-secret. As to the question what should be the correct label for James Bound becomes what should be the proper label for the *name* of the document. I believe this diffuse the confusing.

### 6.2. Context Labeling and Aggregation

Another reason for giving a data two labels is the so called context labeling. A piece of data is insensitive, yet because of its context, it become sensitive. So that piece of data should receive two labels. This is wrong approach, again it leads to logical contradiction. Our classification scheme will label (1) the piece of data unclassified, but (2) the aggregate, which is the data together with its context, top-secret. Note that in data modeling, the context is also expressible by a set of data.

### 6.3. Trusted Subjects

The trusted subjects include the downgrading operation, information flowing downward, which leads to inconsistent with the constraint of the *-property. To avoid logical inconsistency, trusted subjects should not be included in the MLS data model, they should be handled outside of the model [Lin93]

## 7. THE STRUCTURE OF SECURITY CLASSES

Information in databases are represented by views (view instances); a user get his inforamtion by looking at view instances. Views are composite objects of primitive data. So we need structure or algebra to handle the security labeling of composite objects. A relation(or views) scheme is defined by attribute names. The security class of relation scheme or its name can then be derived from the security classes of its attribute names. A relation(or view) instance can be generated from elements (see [Lin92d,e,f] by set representation). The security classes of intensional and extensional objects can in general be derived from the labels of its primitive data. If the security semantics of composite data implies differently, then that is the problem of aggregation [Lin89a, 90b].

## 8. ELEMENT LEVEL LABELING

The so called element level labeling refers to the case of labeling of the elements only (no other data are labeled). It is different from our labeling; we label elements as well as tuples, views and etc..

The meaning of element labeling seems different from systems to systems. One proposal is that (1) the label of an element is labeling the association between primary key and the elements, and (2) the label of the primary key is labeling the existence of the tuple or entity. This approach has several deficiency.

8.1. Incomplete Classification: This proposed approach is somewhat naive. A relation is a very complex mathematical object. For example, let us consider the relation of Example 1. the relation RECORD has the following mathematical sub-structures

(1) There are six associations (binary relationship):

| | |
|---|---|
| Name CL1 | Salary CL2 |

| | |
|---|---|
| Name CL1 | Telephone CL3 |

| | |
|---|---|
| Name CL1 | Occupation CL4 |

| | |
|---|---|
| Salary CL2 | Telephone CL3 |

| | |
|---|---|
| Salary CL2 | Occupation CL4 |

| | |
|---|---|
| Telephone CL3 | Occupation CL4 |

(2) There are four ternary relationship

| | | |
|---|---|---|
| Name CL1 | Salary CL2 | Telephone CL3 |

| | | |
|---|---|---|
| Name CL1 | Salary CL2 | Occupation CL4 |

| | | |
|---|---|---|
| Name CL1 | Telephone CL3 | Occupation CL4 |

| | | |
|---|---|---|
| Salary CL2 | Telephone CL3 | Occupation CL4 |

(3) One quaternary relatioship (the whol relation)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Name | CL1 | Salary | CL2 | Telephone | CL3 | Occupation | CL4 |

Each relationship potentially has its own semantics, and is, in general, independent from each other. However, in this approach, only the first four relationships (out of 11 relationships) are labeled. Could this four labels represent the 11 semantics? We would say no. This is similar to the well-known "connection trap" where people tend erroneously to use binary relationships to construct a ternary relationship [Date91].

In general, this proposal uses a subset of associations to represent the security semantics of the whole relation. So the classification can not be complete, we beleive many relationships (informations) are not protected.

## 8.2. Other Specific Problems

*All key relations*

Under this proposal, the security labeling will lost its power if the relation is the so called all key relation(i.e., every attribute is participated in the primary key). The relationships between attributes are not classified.

*Relational operations and reorganization of databases*

When we join two relations A, B to get a new relation C, the primary key of C usually can not be the old primary keys of A or B. Since C has new a primary key, the binary relationship between an element and the primary key changes, so the label of elements needs to be changed too. So we have to re-label the whole new relation, element by element, whenever we conduct any relatinal operation. In practice, a DBA often has to restructure the database, because of the practicle performance problem. Each reorganization changes the meaning of security labeling. Therefore the whole database has to be re-labeled again. This appraoch is not acceptable.

## 9. TUPLE LEVEL LABELING

The so called tuple level labeling refers to labeling the tuples in base relations, but nothing else. It is different from our labeling; we label elements as well as tuples, views and etc..

Incomplete Model Specification.

These models do not specify their rules of labeling new tuples created by any relational operation. There are two possible default assumptions:

(1) the high-water-mark policy, and
(2) the least upper bound policy.

These two assumptions appear to be similar, but actually they are fundamentally different.

(1) High-water-mark policy: The label of derived data (e.g., the new tuple) are assigned by implementations. In other words, it get the highest label it touched. So a piece of derived data may reach the screen with different labels, because the label of a derived data depends on how the data were traveled through the system. A query optimize may not choose the same path (because of some other users) to answer the same query. This may result in a serious miss judgment. For example, suppose the returned tuple from a query is the tuple (Smith, Mission Taho). Whether the tuple is labeled top-secret or unclassified may lead to different impression to a user, which could lead to disastrous decision.

(2) Least upper bound policy. Under this policy, if a user can see individual tuple of a set, then he can see the whole set. In other words, there is no aggregation problem in tuple level, because the set of tuples will receive the least upper bound. However, at the element level the aggregation problem could exit. Such a near inconsistency assumptions on the model is not very healthy. The so called on second path analysis are based on such implicit assumptions

Both default assumptions have some deficiency.

## 10. CONCLUSION

Recently DOD seems come to a conclusion that provable security system is impossible. We will not dispute the conclusion itself. But we do want to remark that the conclusion may be based on inaccurate studies. We hope DoD will keep the question open until more studies are conducted.

## REFERENCES

[Burns90] Rae Burns, Integrity and Secrecy: Fundamental Conflicts in the Database Environment, Proceedings of The Third RADC Database Security Workshop, June 5-7, 1990, pp. 37-40.

[Date81,86,90] C. Date, Introduction to Database Management Systems, Addison-Wesley, 1981,86,90.

[Denn76] D. E. Denning. "A Lattice Model of Secure Information Flow", Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236 - 243.

[Elmas94] Elmasri and Navathe, Fundemental of Database Systems, Benjamin/Cumming Inc., 1994

[Land82] Carl Landwehr, Formal Model fro Computer Security, ACM Computing Surveys, September, 1981, pp.247-278.

[Lin92a] T.Y. Lin. Aggregation and Fuzzy Sets, Fifth Rome Laboratory Database Security Workshop, October 4-7, 1992

[Lin92c] T. Y. Lin, Attribute Based Data Model and Polyinstantiation, Sept 7-11, Madrid, Spain, IFIP Congress, 1992

[Lin92d] T.Y. Lin, Inference Secure Multilevel Databases, Proceeding of IFIP WG11.3 Workshop on Database Security, August 18-22, 1992.

[Lin92e] T. Y. Lin, The World Model and Polyinstantiation, TR December 1992

[Lin92f] T. Y. Lin, Inferences in Multilevel Databases, TR December 1992

[Lin91] T.Y. Lin, "Inference" free multilevel database system, Proceeding of the Fourth RADC Database Security Workshop, Providence, Little Compton, RI, April, 1991.

[Lin90a] T.Y. Lin, Probabilistic Measure on Aggregation, Proceeding of 6th Annual Computer Security Application Conference, December, 1990.

[Lin90b] T.Y. Lin, Multilevel Database and Aggregated Security Algebra, Database Security: Status and Prospects IV, edited by S. Jajodia and C. E. Landwehr, North Holland, 1991 (Final Revision of Database, Aggregation and Security Algebra (Lattice), IFIP WG11.3 Workshop on Database Security, Sept. 1990.

[Lin89a] T.Y. Lin, Commutative Security Algebra and Aggregation, Research Direction in Database Security, II, Proceedings of the Second RADC Workshop on Database Security, December 22, 1989.

[Lin89b] T.Y. Lin, A Generalized Information Flow Model and Role of System Security Officer, Database Security: Status and Prospects II, edited by C. E. Landwehr, North Holland, 1989.

[Lin89c] T.Y. Lin,16. Security Algebra and Formal Models, Proceedings of IFIP WG11.3 Workshop on Database Security, September 5-7, 1989 (with L. Kerschberg and R. Trueblood, and final revison appear at Database Security: Status and Prospects III, edited by D. Spooner and C. E. Landwehr, North Holland, 1990.

[Lunt90] T. F. Lunt and Donovan Hsieh, "SeaView Secure Database System, A Progress Report", Proc of European Symposium on Research on Computer security, France, October, 1990.

[Lunt89] T. F. Lunt, "The True Meaning of Polyinstantiation", The Proceedings of The Third RADC Database Security Workshop, June 5-7, 1990, pp.26-36.

[Mai83] D. Maier, The theory of Relational Databases, Computer science Press, 1983.

[TsLo82] D. C. Tsichritzis and F. H. Lochovsky, Data Models, Prentice-Hall, Englewood Cliffs, N.J., 1982

# Towards A Policy Framework for Multilevel Databases*

Xiaolei Qian
Computer Science Laboratory, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025

## 1 Problem Statement

As more multilevel databases are built and connected through computer networks, a wide variety of secure data sources will become accessible. A big challenge presented by this technology is the secure interoperation of multilevel databases containing data with mismatched security policies. Providing secure interoperation of multilevel databases not only makes it possible to reliably share data in isolated military and civilian databases, but also increases users' confidence and willingness in such sharing.

As a prerequisite to the secure interoperation of multilevel databases containing data with mismatched security policies, the security policies of component databases, as well as the potential mismatches between them, have to be precisely characterized. Existing literature has been vague on what constitutes a security policy, its content ranging from high-level specifications such as the type of access control (mandatory or discretionary access control) or the kind of model (noninterference or Bell-LaPadula), to designer's belief or preferences such as whether polyinstantiation is allowed, to low-level specifications such as the number of levels and categories allowed in a lattice. A formal policy framework is needed within which security policies could be characterized and compared [6].

It has been widely accepted that a mandatory access control (MAC) policy consists of four components: a set of subjects, a set of objects, a lattice, and a mapping that associates levels in the lattice to subjects and objects [9]. This works well for multilevel operating systems, because objects such as files do not carry semantics. For multilevel databases where data carry semantics, the same mapping of levels to objects such as elements in tuples could have completely different meanings [19]. For example, consider a relation SMD(Starship, MId,

*This work was supported by U.S. Department of Defense Advanced Research Projects Agency and U.S. Air Force Rome Laboratory under contract F30602-92-C-0140.

Destination). A secret label on element Rigel of tuple (Enterprise, 101, Rigel) in SMD could mean that the fact "Enterprise is going to Rigel" is secret, or the fact "some starships are going to Rigel" is secret, or even the word "Rigel" is secret. This confusion suggests that something critical is missing with the traditional formulation of MAC policies in multilevel databases, namely the semantics of object labels. This problem is crucial in the secure interoperation of multilevel databases. For example, if the secret label on Rigel means that the fact "some starships are going to Rigel" is secret in database A, and means that the word "Rigel" is secret in database B, then unclassified users could query all existing destinations in database A and obtain "Rigel" through interoperation with database B. The canonical MAC policy for federated databases proposed in [13] does not solve this problem.

The formulation of a MAC policy in a multilevel database often includes some constraint policies, such as the labeling policy of Seaview [11] and the classification constraints of LDV [5]. Constraints are the most important means of specifying data semantics. However, existing multilevel databases provide neither a precise definition of constraint validity nor an efficient mechanism of constraint enforcement. In fact, it has been argued [1, 2, 12] that integrity enforcement is in fundamental conflict with secrecy enforcement: no multilevel databases could simultaneously satisfy both integrity and secrecy requirements.

An important characteristic of MAC policies is the upward information flow in the lattice, which indicates the believability of low data at high levels. For multilevel operating systems where objects do not carry semantics, low data are always believed at high. For multilevel databases where data carry semantics expressed by constraints however, low data could contradict high data. For example, if we require that high SMD tuples have unique MId elements and (Enterprise, 101, Rigel) is a high tuple in SMD, then the low tuple (Enterprise, 102, Rigel) in SMD could not be believed at high. This problem suggests that upward information flow should be constrained in the formulation of MAC policies in multilevel databases.

Constraints also bring about the danger of inference channels. Inference channels could be obtained either by knowing the constraints enforced by a database or by observing the behavior of a database in enforcing the constraints. For example, consider another relation MT(MissionId, Type). If we require that every MId element in relation SMD refers to a MissionId element in MT, and a low MId element refers to a high MissionId element, then low users could infer the existence of the high MissionId element. If we require that every high MId element in SMD refers to a low MissionId element in MT, then the attempt to delete a low MissionId element referred to by a high MId element would either cause a loss of high data or enable low users to infer the existence of the high MId element. Thus the formulation of MAC policies in multilevel databases should provide additional means to detect and remove such inference channels.

## 2 Our Policy Framework

We restrict ourselves to multilevel databases whose MAC policies have the simple security property and the ∗-property of the Bell-LaPadula model, which ensure that information does not flow downward in the lattice.

- *The Simple Security Property* A process is allowed a read access to a tuple only if the former's clearance level is identical to or higher than the latter's classification level in the lattice.
- *The ∗-Property* A process is allowed a write access to a tuple only if the former's clearance level is identical to or lower than the latter's classification level in the lattice.

Our formulation of a MAC policy in a multilevel database has seven components:

1. a lattice,
2. a set of subjects,
3. a set of objects,
4. a mapping of subjects and objects to levels in the lattice,
5. an interpretation policy,
6. a view policy, and
7. an update policy.

The first four components together correspond to the traditional formulation of MAC policies in multilevel operating systems. In the rest of the paper, we discuss examples of the last three components of our policy framework, using the lattice in Figure 1 and the schema in Figure 2.

## 3 Interpretation Policy

An interpretation policy maps a multilevel database to a multilevel theory. Through this policy, the superficial syntactic difference in object labels is abstracted away, and the semantic difference hidden in object labels is made precise. As a consequence, the interpretation policy makes it possible to compare the semantics of multiple MAC policies.

For example, suppose that a high label on element `spy` in tuple `(Enterprise, spy, Rigel)` means in one database that low users should not know that "Enterprise is on a spy mission", but means in another database that low users should not know that "there is a

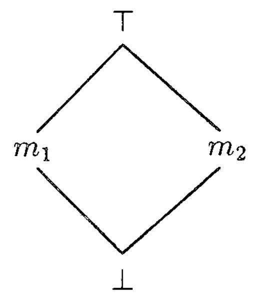


Figure 1: A Lattice

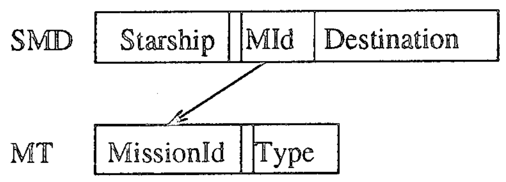


Figure 2: A Schema

starship on a spy mission". The semantic interpretation would map the high label to the high formula $(\exists x)$SMD(Enterprise, spy, $x$) for the first database, and to the high formula $(\exists x, y)$SMD(Enterprise, $x, y$) for the second database. By comparing these two formulas, we can infer that the second database has a weaker security policy about Enterprise than the first database, because it protects less high information.

As case studies, we have developed natural interpretation policies for multilevel relational databases with tuple-level and element-level labeling respectively, which have properties that are commonly recognized as desirable. Based on these policies, we have identified practical design trade-offs in choosing between tuple-level and element-level labeling [17].

## 4 View Policy

The simple property of the Bell-LaPadula model gives the *visibility* requirement on what low data are visible at high. As we pointed out above, visibility should be distinguished from the *believability* requirement on what low data are believed at high in order to avoid inconsistency. A view policy states this believability requirement for a set of integrity constraints.

The filter function [7, 8, 10] and the security logic [3] proposed in the literature take one extreme position by equating believability to visibility, thus maximizing believability. However, integrity is compromised if a low tuple contradicts some high tuples with respect to the constraints, which leads to an invalid high database. For example, consider the following multilevel relation over the lattice of Figure 1 and the schema of Figure 2:

| Starship | Mission | Destination | |
|---|---|---|---|
| Enterprise | $\angle$ | Talos | $\top$ |
| Enterprise | 102 | Rigel | $m_1$ |
| Enterprise | 103 | Rigel | $m_2$ |

When querying the mission of Enterprise at level $\top$, users will get back both 102 and 103, which contradicts the constraint "starships have unique missions".

Smith and Winslett proposed a belief-based semantics of the multilevel relational model [20], which defines a multilevel relational database as a set of unrelated single-level relational databases, one for every level. They made a clear distinction between visibility and believability, and took the other extreme position by allowing no low tuples to be believable at high, thus minimizing believability. Their semantics serves as a nice framework within which other semantics could be compared. However a multilevel relational database that directly employs their semantics would no longer be multilevel — it would be a set of single-level relational databases in which there is no upward information flow across levels. For example, consider the following multilevel relation over the lattice of Figure 1 and the schema of Figure 2:

| Starship | Mission | Destination | |
|---|---|---|---|
| Enterprise | 102 | Rigel | $\perp$ |

When querying the mission of Enterprise at level $\top$, users will get back an empty answer, because no information about Enterprise is considered believable at that level.

Thuraisingham first formalized the distinction between visibility and believability by a proof-theoretic semantics of the multilevel relational model [21], which consists of a nonmonotonic inference rule stating that low data are believable at high as long as they do not contradict high data. Given two low tuples labeled incomparably, what happens if either tuple does not contradict high data, but their combination does? To determine what is believable at high, the result of Thuraisingham's approach would depend on the (random) order in which the nonmonotonic inference rule is applied to these two tuples, which introduces ambiguity. For example, consider the following multilevel relation over the lattice of Figure 1 and the schema of Figure 2:

| Starship | Mission | Destination | |
|---|---|---|---|
| Enterprise | 102 | Rigel | $m_1$ |
| Enterprise | 103 | Talos | $m_2$ |

When querying the mission of Enterprise at level $\top$, users will get back either 102 or 103 but not both. It should be noticed that such problems occur even with a totally ordered security lattice, if we allow arbitrary constraints. For example, a constraint could state that there should be no more than two starships going to Rigel. If we have one high tuple (Enterprise, 101, Rigel) together with two low tuples (Voyager, 102, Rigel) and (Discovery, 103, Rigel), then at most one low tuple is believable at high, but it is unclear which one should be.

A view policy should have three desirable properties:

1. it ensures the validity of constraints,
2. it maximizes upward information flow, and
3. it is deterministic.

As a case study, we have developed a view policy for multilevel relational databases with tuple-level labeling, where the constraints consist of key-based functional and referential dependencies, which has all the desirable properties identified above [16].

## 5 Update Policy

An update policy specifies the enforcement of a set of constraints in performing a set of updates, such that inference channels are eliminated.

Let us consider the restricted-value policy of [18] and the insert-low policy of [22], both of which are designed to eliminate inference channels in the enforcement of the no-polyinstantiation constraint. For easy presentation, we adapt these policies to the context of multilevel databases with tuple-level labeling. The no-polyinstantiation constraint states:

*Two distinct tuples cannot have identical primary key values.*

If low users insert a tuple which has the same primary key value as an existing high tuple, then either the low insertion has to be rejected, leading low users to infer the existence of the high tuple, or the high tuple has to be overwritten, causing a loss of high data. Similarly, if high users insert a tuple which has the same primary key value as an existing low tuple, then either the low tuple has to be deleted, leading low users to infer the existence of the high tuple, or the high insertion has to be rejected, causing a loss of high data.

The example below illustrates how the restricted-value policy removes this dynamic inference channel in the no-polyinstantiation constraint. Consider the following multilevel relation over the lattice of Figure 1 and the schema of Figure 2:

| Starship | Mission | Destination | |
|---|---|---|---|
| Enterprise | 102 | Rigel | $\perp$ |

When users try to replace 102 by 101 at level $\top$, the update is extended to:

1. Replace 102 by $\surd$ at level $\perp$.

2. Insert (Enterprise, 101, Rigel) at level $\top$.

The extended update ensures no-polyinstantiation at the price of introducing a (partial) static inference channel, because users at level $\perp$ can infer from the restricted-value $\surd$ that Enterprise has a high mission. Moreover, the high update is extended with a low insertion, which is against the spirit of the $*$-property of the Bell-LaPadula model.

The example below illustrates how the insert-low policy removes this dynamic inference channel in the no-polyinstantiation constraint. Consider the following multilevel relation over the lattice of Figure 1 and the schema of Figure 2:

| Starship | Mission | Destination | |
|---|---|---|---|
| Enterprise | 101 | Rigel | $\top$ |

When users try to insert tuple (Enterprise, 102, Rigel) at level $\perp$, the update is extended to:

1. Delete (Enterprise, 101, Rigel) at level $\top$.

2. Insert (Enterprise, 102, Rigel) at level $\perp$.

The extended update ensures no-polyinstantiation at the price of losing high data.

An update policy should also have three desirable properties:

1. it does not introduce inference channels,

2. it does not affect data at lower or incomparable levels, and

3. it does not cause data loss at higher levels.

Based on these properties, we have provided practical design guidelines for the appropriate specification of constraints, whose enforcement would not jeopardize secrecy requirements [15]. We have also developed an update policy for multilevel relational databases with tuple-level labeling where the constraints consist of polyinstantiation and referential security properties, which has all the desirable properties identified above [14].

## 6 Conclusion

We have used our policy framework to compare the MAC policies commonly imposed in or proposed for multilevel databases. The comparison makes it clear that the space of MAC policy mismatches between heterogeneous multilevel databases is significantly larger than the space of semantic mismatches between heterogeneous single-level databases. Our framework could be used to capture and resolve the MAC policy mismatches in the interoperation of heterogeneous multilevel databases. As an initial step in this direction, we have investigated the secure interoperation of multilevel databases whose MAC policies mismatch in the lattice component [4].

## References

[1] R. K. Burns. Referential secrecy. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 133–142, 1990.

[2] R. K. Burns. Integrity and secrecy: Fundamental conflicts in the database environment. In *Proceedings of the Third RADC Database Security Workshop, Technical Report MTP 385, MITRE*, pages 37–40, 1991.

[3] J. Glasgow, G. MacEwen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.

[4] L. Gong and X. Qian. The complexity and composability of secure interoperation. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 190–200, May 1994.

[5] J. T. Haigh, R. C. O'Brien, and D. J. Thomsen. The LDV secure relational DBMS model. In S. Jajodia and C. Landwehr, editors, *Database Security, IV: Status and Prospects*, pages 265–279. North-Holland, 1991.

[6] H. H. Hosmer. Integrating security policies. In *Proceedings of the Third RADC Database Security Workshop, Technical Report MTP 385, MITRE*, pages 169–173, 1991.

[7] S. Jajodia and R. Sandhu. Polyinstantiation integrity in multilevel relations. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 104–115, 1990.

[8] S. Jajodia and R. Sandhu. Toward a multilevel secure relational data model. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 50–59, 1991.

[9] C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–278, September 1981.

[10] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. The Seaview security model. *IEEE Transactions on Software Engineering*, 16(6):593–607, June 1990.

*[11] T. F. Lunt, P. G. Neumann, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. Secure distributed data views: Security policy and interpretation for DBMS for a class A1 DBMS. Technical Report RADC-TR-89-313, Vol. 1, Rome Air Development Center, Air Force Systems Command, December 1989.

[12] C. Meadows and S. Jajodia. Integrity versus security in multilevel secure databases. In C. Landwehr, editor, *Database Security: Status and Prospects*, pages 89–101. North-Holland, 1988.

[13] G. Pernul. Canonical security modeling for federated databases. In *Proceedings of the IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems*, 1992.

[14] X. Qian. A model-theoretic semantics of the multilevel secure relational model. Technical Report SRI-CSL-93-06, Computer Science Laboratory, SRI International, November 1993.

[15] X. Qian. Inference channel-free integrity constraints in multilevel relational databases. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 158–167, May 1994.

[16] X. Qian. A model-theoretic semantics of the multilevel relational model. In M. Jarke, J. Bubenko, and K. Jeffery, editors, *Advances in Database Technology — EDBT'94, Lecture Notes in Computer Science 779*, pages 201–214. Springer-Verlag, March 1994.

[17] X. Qian and T. F. Lunt. Tuple-level vs. element-level classification. In B. M. Thuraisingham and C. E. Landwehr, editors, *Database Security, VI: Status and Prospects*, pages 301–315. North-Holland, 1993.

[18] R. Sandhu and S. Jajodia. Eliminating polyinstantiation securely. *Computers & Security*, 11:547–562, 1992.

*Although this report references the limited document noted above, no limited information has been extracted. Document is limited to DOD and DOD contractors only; critical technology; Dec 1989.

[19] G. Smith. Modeling security-relevant data semantics. *IEEE Transactions on Software Engineering*, 17(11):1195–1203, November 1991.

[20] K. Smith and M. Winslett. Entity modeling in the MLS relational model. In *Proceedings of the Eighteenth International Conference on Very Large Data Bases*, pages 199–210, 1992.

[21] B. M. Thuraisingham. A nonmonotonic typed multilevel logic for multilevel secure database/knowledge-base management systems. In *Proceedings of the Fourth IEEE Workshop on Computer Security Foundations*, pages 127–138, 1991.

[22] S. R. Wiseman. Control of confidentiality in databases. *Computers & Security*, 9(6):529–537, October 1990.

## Discussion: Models

**Discussion Leader: Marvin Schaefer, ARCA Systems, Inc.**

(paper not available)

# OBJECT-ORIENTED DATABASE MANAGEMENT

# Discussion of a New Secure Object-Oriented Data Model

William R. Herndon

The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
wherndon@mitre.org

## 1 Introduction

Object-oriented database management systems (OODBMSs) are gaining popularity due to their inherent ability to represent conceptual entities as objects, paralleling the way humans view the his power of representation has led to a new generation of object database managers that can support applications such as computer aided design and computer aided management (CAD/CAM), multimedia information processing, artificial intelligence, and process control. However, these increasingly popular systems do not provide adequate support for security of operation. That is, OODBMSs need to operate securely, often in a multilevel fashion, in order to overcome both malicious corruption of data and to prevent unauthorized access to and use of classified data. Consequently, multilevel database management systems are needed in order to ensure that users cleared to different security levels access and share a database with data at different security levels in such a way that they obtain only the data classified at or below their level. Such database systems are called multilevel secure (MLS) OODBMSs.

## 2 Background

Recently, several approaches to securing OODBMSs have been proposed. While these efforts have made valuable contributions, some major issues remained unexamined, including:

- *Consistency with industry trends*: For greater acceptance by users and vendors we maximize consistency with de facto standards like C++ and the work of the Object Management Group. Some existing models forbid multiple inheritance, make method call a very expensive operation, or tie security to the language's encapsulation (which C++ allows programmers to circumvent).

- *Model flexibility:* In the absence of installed products and applications, it is difficult to distinguish which features and freedoms will be essential. Each model restriction (e.g., requiring that all attributes of an object be at the same security level) carries a major risk: It may later be found very harmful, and impractical to remove from the implementation. Therefore, it seems desirable to seek generality.

In a similar vein, some models impose restrictions to rule out states that appear unreachable or unnecessary (such as an instance whose key is classified higher than other attributes). Paradoxically, such restrictions often complicate the model and implementation, since they must be documented and enforced. Also, in special circumstances (created by privileged operations) such states can be very useful.

- *Element-level access control:* Our extensive analysis conducted at MITRE concluded that applications should be written in terms of the natural conceptual objects of the problem domain, and that these objects could include elements (i.e., attribute values) at multiple security levels. Many of the existing models provide protection at the object level. We also concluded that for the combination of (application + DBMS), neither semantic clarity, assurability, nor performance need degrade. Our model therefore permits each attribute of each object to be labeled, independent of all other labels.

- *Treatments of polyinstantiation:* Existing object models provide a limited treatment of conflicting data. Existing relational models either do not provide clear semantics for such data (in terms of statements about the outside world) or else provide too much semantics, making them well suited to some applications but ill suited to others.

- *Collections of data:* Collections (e.g., sets, lists, trees) constitute the major means by which a database organizes large amounts of information. Existing secure relational models support one kind of collection, the relation as a set of tuples. Commercial object models support additional structures. However, collections appear not to be included in any published secure object model.

## 3 UFOS: A New Model

Due to the problems discussed above, we have created a new model, called Uniform Finegrained Object Security (UFOS, pronounced U.F.O.s).

Secure OODBMSs are likely to combine technology specifically developed to secure OODBMSs with technology from contemporary secure relational DBMSs. Therefore, in an earlier work we have compared the problems and solutions of the object and relational worlds. We also examined the security impact of capabilities unique to OODBMSs. We have emphasized methods of providing security to the base of *mainstream* OODBMS technologies, because object models and implementation techniques that are unique to the security community will lead to systems having functional capabilities that substantially lag behind that of mainstream systems and will too often be unacceptable to users. Thus, the UFOS model has, as its basis, the following characteristics:

- A labeling scheme that is uniform throughout the model with respect to both what is labeled and how it is labeled.

- Support for object collections

- Support for multilevel conceptual objects

- A nonrestrictive model for class hierarchies and inheritance that circumvents monotonicity restrictions

- An model layer, existing above the base model, that supports the definition and implementation of polyinstantiation semantics (the *polystrorage layer*)

Work to date includes the development of the basic and advanced data models and the examination of the impact of security on the data model and the development of applications. Currently, we are examining a number of commercial OODBMSs with the intent of identifying the most appropriate product to host our model. We have developed criteria for evaluating the products that require a close examination of the products' underlying data model and architecture.

## 4 Model Utility

Current work on this task has focused on demonstrating the utility of our model by applying it to a sample set of Navy applications that require support for multilevel operation as well as a flexible fine-grained access control structure. These "envisioned" applications include logistics support systems, maintenance information systems, and combat information systems. All of these have need of DBMS services and can benefit in a number of ways from MLS operation. In fact, some of the scenarios that have been developed, point to the need for an integrated MLS database, possibly extending over many heterogeneous DBMSs and systems. In such an environment, the novel aspects of the UFOS model provide some distinct advantages.

For example, a maintenance information system for Navy aircraft, can benefit from all of the model facets listed above: fine-grained labeling, collections, and polystorage. In addition to direct support for querying over composite objects (made possible by fine-grained labeling), a system based on the UFOS model could also be used to support sophisticated level-based training simulations, where information appropriate to a given technicians clearance level could easily be made available. Likewise, schematics and component lists could be tailored to fit the level of a user through the use of polystorage and secure collections. In addition, since maintenance personnel cannot always be shielded from the knowledge of the existence of certain components, cover stories can be inserted in the database to further enhance security.

# Trusted ONTOS Prototype
Preliminary Considerations

Marvin Schaefer
ARCA Systems, Inc.
Columbia, MD

Sandra Wade
ONTOS, Inc.
Vienna, VA

## 1. Project Overview & Scope

In the late summer of 1993, the authors[1] began research under contract to the National Security Agency and Rome Laboratory to begin development of an informal access control model [1] for a trusted object-oriented database management system (ODBMS). This study is intended to serve as the basis for future efforts to produce a trusted prototype of an ODBMS offering features comparable to those required for Class B1 of the DoD's Trusted Computer System Evaluation Criteria (TCSEC) and the associated Trusted Database Interpretation (TDI) of the TCSEC.

The philosophy behind object oriented technology is becoming the *de rigueur* standard for the industry, even though there is presently no universal model that serves as a standard for individual ODBMS implementations. Several ODBMS products are currently serving a growing user community. They are being used with greater frequency by the government and industry because they offer many benefits over existing technologies such as increased performance for complex applications, support for unusual data types, and a highly flexible data model. Additionally, with the reduction of budgets in both the government and industry, object-oriented technology is gaining a wider audience for its potential to reduce overall life cycle costs by enabling component based software development, promoting software re-use, and supporting extensible solutions. It is evident that ODBMS technology will be the basis for future DoD database applications. There is a clear need for a high integrity, multilevel secure, ODBMS.

Although there have been numerous paper studies, there are presently no *worked* examples of a trusted ODBMS, extant or under development. It is equally important to note that although the more traditional concepts and architecture of relational DBMS (RDBMS) tend to dominate the TDI, there are no *interpretations* of how specific TCSEC requirements are to be applied to an ODBMS. The present effort is intended to support future research and development needed in order better to understand a) the security related issues in the design and implementation and b) the evaluation, and especially the *assurance* requirements for a high-integrity, multilevel secure ODBMS that offers B1 features.

This study is intended to take a fresh look at the trusted DBMS problem. Previous, relational model-based approaches, have largely been based on a set of security architectures that lead to

[1]Marv Schaefer was affiliated with CTA Incorporated at the time.

polyinstantiation or selective database replication as a means of preserving confidentiality. However, use of this strategy is often at the cost of database consistency, integrity, performance, and the ability to see updates without delay. Further, the semantics and operational consequences of polyinstantiation have sometimes proven to be inadequately understood by users and have resulted in database update inconsistencies. Given that object-oriented architectures invite the introduction of new security architectures, the opportunity is present to re-examine alternatives that could result in a more favorable tradeoff between the objectives of confidentiality and database integrity.

## 2. Approach

The approach taken in this study is based on a survey of relevant prior research in DBMS security, with a concentration in object-oriented studies. For the most part, these consist of formal and informal models and descriptions of hypothetical implementation strategies. Some of the literature surveyed in the study identified specific constraints on the model and on the resultant functionality that follow from sometimes identified aspects of the access control model or the envisioned evaluation class. For example:

- B2 and higher evaluation classes concentrate heavily on issues of minimality in the TCB, least privilege, and covert channels. These concerns tend to force the policy and design into directions taken by SeaView, LOCK DataViews, etc. — so that potential channels are reduced through the introduction of polyinstantiation or data replication at different classification levels. This is done to preclude inferential attacks that may otherwise disclose sensitive information if known multilevel integrity constraints were to be probed by a knowledgeable adversary.

- Least privilege considerations can interfere with the ability of a trusted DBMS to detect cases in which inadvertent polyinstantiation or breaches in referential integrity have transpired. This is particularly the case when a user logged in at a level lower than "database-high" performs updates on the database. This is because the trusted DBMS runs as a subject at the user's login level and its associated privileges, and cannot see any of the database or metadata not dominated by that subject's need-to-know and clearance level. Since the DBMS cannot, under these circumstances, generally obtain a complete and consistent view of a multilevel database, it is generally incapable of managing all aspects of the data model itself.

- The above assurance considerations also have their affect on concurrency and transaction management. Contemporary user requirements call for DBMSs that support multiple concurrent users and preserve transactional integrity. This topic is one of complexity and intensive continuing research in the untrusted community. It only becomes more complex and less certain when covert channel-free confidentiality requirements are imposed.

The above constraints and complexity have largely come from an evolutionary approach to DBMS security that is based on prior results from operating system security and initial attempts to retrofit relational DBMSs onto trusted operating system architectures. The resultant trusted DBMS policies have therefore largely been constrained by the intrinsic limitations of operating system policy that could not be modified without placing an underlying trusted operating system's TCSEC assurances *and* evaluation rating in jeopardy.

ODBMSs are evolutionary with respect to object-oriented operating systems which are largely client-server based. Resulting experience, e.g., Trusted Mach, has shown that these latter architectures do not evaluate readily against the established TCSEC requirements; the latter frequently require extensive interpretation in order to be applied to contemporary architectures. We have concluded that much of the evaluation difficulty is not so much caused by inherent weaknesses in an object-oriented operating system security architecture as by the mismatch between the conceptual models on which the two syndromes are based.

Therefore, it would be productive to return to First Principles and begin with the definition of a desired set of control objectives and database properties. Since modeling is the first step to be taken in building the foundation for a proof of concept prototype, many of the decisions regarding our policy model have been made with the goal of eventual implementation. We consider it extremely important that the resulting policy model be amenable for refinement into a viable commercial trusted product. Because of our familiarity with the ONTOS ODBMS architecture and internals, we have systematically assessed concepts against the realities of modifying this product to support the multilevel ODBMS model.

The proposed methodology for analysis is (a) to hypothesize a complete multilevel data model including labeled database entities, (b) to establish through informal analysis that the model is internally consistent, (c) to superimpose access rules onto this model, (d) to analyze the adequacy of the model against multiuser database goals and objectives, (e) to establish the existence of an acceptable implementation strategy, and (f) to study the expected security properties of the resultant constrained abstract "design". Clearly, considerable iteration is required in the above. We believe that this approach will yield a usable combination of an access control and integrity model, a hypothetical security architecture, and an interpreted set of criteria against which to measure any mathematically faithful implementation.

During the course of this study, the relevant literature was surveyed extensively. In particular, we consulted the proceedings of all major DBMS security conferences and workshops, and both published and internal reports from institutions conducting research in the field. We also conducted numerous discussions with members of the trusted DBMS research community. The Annex contains a partial bibliography of materials used in this study.

The survey of existing literature took place in several passes. The first pass of the literature was intended to identify common objectives and common restrictions. The second pass was engaged to identify approaches and issues relevant to an object-oriented data model versus a relational model. Finally, a third pass was conducted to identify those approaches appropriate to a C++ environment as opposed to Smalltalk (message passing) environment. The last two passes are discussed in more detail below.

While it appears possible to propose a straight-forward mapping between the relational data model and the object model, and therefore apply an access control policy designed for relational databases to object databases, the resultant model falls short in securing all aspects of the object database. Because the object model subsumes the relational data model, not only does the access control policy need to secure the data in an object database, the policy also needs to address issues such as: how to handle inheritance in the data model; how to handle references between objects; how to secure methods; and how to handle iteration over groups of objects stored in the same aggregate. Unfortunately, much of the literature approached the problem of defining a multilevel security policy for ODBMSs by relying on existing work on relational databases. Because we were concerned with all aspects of the object model, much of this literature was consulted, but only selectively considered, as a foundation from which to define our working model.

Additionally, much of the literature that focused on the object model in its entirety discussed message filtering approaches to multilevel ODBMS security which are specific to the Smalltalk paradigm. Although Smalltalk was the predominant object-oriented language when much of the literature in securing an ODBMS was written, for various reasons most of the commercial ODBMS vendors such as ONTOS, Inc., Objectivity, Inc., Object Design, Inc., $O_2$ Technology, POET Software, and Versant Object Technology have written their products using the C++ language. Only Servio Corporation has written its product using the Smalltalk language. While the ability to pass information at runtime as messages between objects exists in Smalltalk, it does not exist in C++. In C++, all "messages" are implemented as methods or procedures that must be defined at compile time. Because there is no central message passing mechanism to control information flow in a C++ environment, the message filtering literature was also excluded as forming the basis from which to define our model.

## 3. Issues in Securing an ODBMS

As mentioned previously, defining a multilevel access control policy for an ODBMS is not as straightforward as applying techniques that have successfully passed evaluation against the DoD Trusted Computer System Evaluation Criteria and associated Trusted Database Interpretation. In this section, we discuss aspects of the object model which are not present in the relational data model and have not been given adequate attention as they relate to security. They include: inheritance, relationships and referential integrity, aggregates, and methods and polymorphism.

### 3.1. Inheritance

In an ODBMS, classes are organized into a hierarchy with each subclass inheriting all of the attributes and methods from its superclass(es). Because metadata information about a particular class is established at run-time, it must be possible for an application to have access to all metadata up its inheritance tree. To do so without potentially illegal inheritance information flows appears to require monotonically non-decreasing class hierarchies. That is, the level of the subtype must always dominate its supertype, i.e., its classification must be no less restrictive than that of the supertype. While this guarantees that information flow down the hierarchy is always confined according to the classification lattice, it does so with the known trade-off of making the data model far more cumbersome for the database designer. The problem is that the data model and the sensitivity of the data are orthogonal; the existence of a subclass does not always imply the need for greater or equal sensitivity. Most models in the literature on multilevel security for ODBMSs enforce the restriction of monotonically non-decreasing class hierarchies. New ideas about dealing with inheritance without this restriction have been discussed in [2].

## 3.2. Relationships and Referential Integrity

Referential integrity is a very important property to establish and preserve in database management. However, it is extremely difficult to find a means to implement truly multilevel referential integrity in a multilevel DBMS. This is because of the need to deny operations that would lead to a violation of referential integrity.

In relational DBMSs, referential integrity guarantees the existence of references to any foreign key. For example, if in a military operations database there is a operation relation that references tanks, there must exist corresponding tank tuples in the vehicles relation. It would be a violation of referential integrity if (a) an operation tuple were created that referenced a specific tank that did not exist in the vehicles relation; or (b) if a tank tuple were deleted from the vehicles relation while there still existed one or more references to it in the operation relation. In the relational model, tests for referential integrity are straightforward and well-defined.

In ODBMSs, references between objects are based on *object-identity* rather than values of keys. Whether or not interobject references are well-defined depends, however, on existential constraints: a referenced object (i.e., referent) must exist for the reference to be valid. Conversely, deletion of a referent object would be dependent on deletion of the reference. Tests for referential integrity are more complex than in the relational context, but can be simplified through provision of bi-directional references as in Morgenstern[3].

In DBMSs that enforce MAC, it is clear that the security level of the reference must dominate the security level of the referent, since otherwise there would be a visible reference to an object that could not be observed from the level of the referencing object. However, it would be impossible to preserve referential integrity unless the security level of the referent object did not also dominate the security level of the referencing object, since attempts to delete the referent must be denied so long as there are any remaining references to it (i.e., the referent must be visible). *Therefore, full compliance with the *-property would lead to the conclusion that the reference and the referent must have equal security levels.* This constraint would be unacceptable for many real-world applications.

There are many situations in which either the reference or referent must have *distinct* security levels. For example, an unclassified vehicle may be necessary for several sensitive military operations. The success of these operations would be jeopardized were needed vehicles to become unavailable without coördination with the operation planners. It could also be compromised were its confidentiality to be prematurely breached. This causes there to be a conflict between the goals of confidentiality and referential integrity. Confidentiality would require that the database always permit the deletion of a vehicle independent of whether it is referenced from a higher level, *even if so doing would compromise referential integrity.* This is because an adversary could infer which vehicles are potentially involved in such operations by sequentially attempting to delete the vehicles from the database and noting which vehicles could not be deleted.

We believe that this problem can be partially addressed at less than the B1 level of assurance by having the DBMS maintain a database high log of violations to referential integrity as they occur. The bi-directional references that simplify enforcing referential integrity will generally involve an implicit breach of the *-property and must be appropriately managed by the TCB and hidden from uncleared users. The proposed model is structured to support the capability to address this problem.

## 3.3. Aggregates

Unlike relational DBMSs, which support only one type of aggregate (e.g., an unordered set of tuples), object DBMSs support a variety of aggregates including Lists, Sets, Dictionaries, and Arrays. While it is possible to apply the same techniques to securing a Table in an RDBMS to a Set Aggregate in an ODBMS, the approach is inadequate for Lists, Dictionaries, and Arrays; all of which possess an implicit or explicit ordering of the objects they contain. Multilevel aggregates, therefore, require a means of interrelating the individual elements of the aggregate without compromising information confidentiality or the basis for labeling. The semantics of how to perform operations on multilevel, ordered-aggregates such as iteration over the members of the list, querying for the cardinality of the list, testing for equality between two lists, querying to see if the list is empty, and copying a list must be researched and identified.

## 3.4. Methods and Polymorphism

Morgenstern [3] and others have introduced the possibility of having classified methods with the additional potential for several distinctly classified instances of a single method to coëxist. An example would be the requirement to support three separate implementations of the Method get_3Dposition. Each implementation would have the identical signature:

[virtual] float get_3Dposition(int *x*, int *y*, int *z*)

The only difference between the three are the actual implementations. The Unclassified version would return an approximation. The Secret version would return a rounded off value. The Top Secret version would return the exact value.

Methods in an ODBMS are typically bound to a single programming language such as C++ or Smalltalk. Many of the research studies in securing a multilevel ODBMS have focused on message filtering approaches [5, 6]. This approach, while applicable to Smalltalk, is not suitable for a C++ language binding. A central mechanism within the TCB, much like the central message passing mechanism within [5, 6], must be identified which can intervene in a C++ environment at run-time to enforce security.

## 4. Objectives for a Multilevel ODBMS

In this section we provide motivation for the decisions made in developing our proposed ODBMS model, and begin with a summary of the goals and objectives we believe are needed to support a trusted ODBMS that offers an integration of multilevel confidentiality and integrity while maintaining a reasonable user interface. It can be argued that the now traditional polyinstantiation approaches meet these objectives and can be used safely by sufficiently sophisticated users. However, because of the subtleties inherent in the semantics of polyinstantiation and the avoidance of covert channels, naïve users are apt to commit errors or misinterpret data. Even sophisticated users may find that their intentions are foiled by the confidentiality mechanisms and the side effects of concurrent use by subjects at differing security levels. We outline some of the motivating difficulties below:

### 4.1. Multilevel Trusted Subjects and Objects

In least privilege architectures, such as those required at and above the B2 level, no portion of the TCB is capable of observing and modifying all of the interrelationships and data values within a multilevel database. The difficulty this causes is that no multilevel update can transpire as an atomic transaction, and concurrent transactions at other security levels may interfere with the intended operation and its consistency. There can be serious problems, even if operating in a single-user environment. For example, if a user operates at a level lower than the most sensitive data, the DBMS's lack of ability to observe all data values and integrity constraints, while serving that user, may result in integrity compromises that cannot easily be detected by any user or DBA, even though they could have been prevented in a commercial single-level environment.

Many of the integrity problems identified above can be eliminated if the DBMS is capable of observing the overall database and its related integrity constraints. However, building a DBMS this way could make it ineligible for B2 or higher levels of trust since it would appear to violate the principles of least privilege, least common mechanism, and TCB minimality, and it could introduce serious covert or inferential channels. The only way in which confidentiality could be preserved would be to implement a TDI Trusted Subject DBMS security architecture. Even with the more relaxed B1 assurance requirements, considerable attention to security-relevant decisions is required in order to ensure that the TCB maintains continuous control over classified data and labeling, and over the interface with the underlying trusted operating system base. This would be defensible only if adequate analysis and confinement were possible. We believe that the model developed in this study serves to justify a credible case that the required assurances can be provided at the B1 level.

### 4.2. Granularity of Labeling

The TCSEC mandates that security relevant decisions be based on the interpretation of sensitivity (i.e., classification and clearance) labels associated with data and with subjects. Using labels to enforce security works well with the classical processes, files, and modes of access that translate into the abstract Read and Write operations of the Bell-LaPadula model. However, it must be borne in mind that while label interpretation is always a syntactic issue, data classification is a semantic issue and cannot, in general, be automated. In particular, the original creation of a file derives its classification, in part, from a conscious and educated decision on the part of its creator; subsequent actions on the operating system file are often safe because of the simplified interpretation of the permitted modes of access. While this is usually an acceptable approach to take in a trusted operating system, it falls short of what is needed in a trusted DBMS because update and transaction semantics are far more complicated and involved.

Issues of implementation often mislead the designer from obtaining a clear understanding of what is needed to fit the requirements of the application, and then considering whether and how it can be represented and enforced. Many researchers have considered the objective of "field-level" granularity of classification, leading to the possibility that the number 17.3 is considered to be Top Secret (Woods Hole, *q.v.*) while 17.4 and 0.1 are unclassified numbers. The point that is being missed is *not* whether 17.4 – 0.1 is Top Secret, but rather what it is that causes 17.3 to have such sensitivity. The reason is always tied to the relationship between the 17.3 and the *unit* or *category* with which it is associated in the real or abstract world of the classification authority. That is, it is not so much the *value* that is classified as the *relationship* between things that is sensitive. Some researchers have based their entire approach to DBMS policy modeling on the notion of data-dependent classification schemes based on data associations [4].

In this study, we concentrated on mechanisms for associating labels with data values as opposed to associating them with the relationship a data value has with its Class, Attribute, or Reference to another object. Our goal was to identify a simple and uniform means of labeling at the finest granularity possible, which could be enforceable at runtime. Equally important was preserving the benefits of the emerging ODBMS model. Based on our overall rationale for classification, the variation on Morgenstern's [3] original concepts of Complete Object and of Object Instance developed into the refined form shown in our model.

## 4.3. System Administration Considerations

Databases, like enterprises, always evolve and undergo modification. Often the modifications are planned, but at times they represent reaction to known compromises to integrity that require rectification. To compensate for inconsistencies that arise from support of a multilevel ODBMS, e.g., accidental polyinstantiation, an appropriate suite of high-integrity utilities needs to be designed and provided.

In this study, considerable attention was focused on the goal of creating a model that would not severely degrade the of ease-of-use of the multilevel ODBMS over a conventional single-level ODBMS. We believe that the present model, which does not explicitly deal with operational matters, will support this objective. In particular, the model is developed with the implicit requirement that the TCB, in its support of users, have full access to all metadata, all data, and all integrity constraints. This provides a planned means for interacting with cleared staff who must define, redefine or repair portions of the database as a consequence of multilevel use.

Currently, the model does not explicitly define any policy-critical personnel rôles or functions. Further examination is required of implementation issues and their interplay before these operational considerations can be made precise.

## 4.4. Polyinstantiation and Data Replication

Many trusted DBMS efforts have several techniques of decomposing a multilevel database into single-level components in order to counter potential covert channels or otherwise disallowed information flows. A common approach in much of the literature is the concept of *polyinstantiation,* with *data replication* as one of the ways in which it may be implemented. In some cases, polyinstantiation is used to provide deliberate, but separate, realities at distinct sensitivity levels — either in the form of disinformation or as plausible cover stories. In particular implementations, however, the device may be either an artifact of updates made by subjects acting at different sensitivity levels, or the basic implementation strategy for a DBMS' security architecture.

In either case, the results of polyinstantion can be contrary to the goals of establishing and preserving database consistency and semantic integrity. Many examples and paradoxes have been presented in the literature that indicate that polyinstantiation is a rich complex that offers both benefits and liabilities. The issues range over the foundations of a consistent data model (e.g., multiple tuples stemming from a single primary key to a relation), how to perform statistical queries on a polyinstantiated database, etc. It appears that each issue can be dealt with individually, but there is not yet an accepted universal theory.

Many investigators have chosen to differentiate between deliberate and accidental cases of polyinstantiation and its manifestation. Deliberate cases include cover stories and corrections performed by subjects cleared to view all of the relevant information. Accidental cases (sometimes called *automatic* polyinstantiation) are those where, as a consequence of updates performed by subjects acting at different security levels, the TCB appears to create object instances that have the same object identity but that otherwise differ in value. The accidental case can result either from the user acting on incomplete information or from vestiges of the *-property. To distinguish all intentional cases from the accidental case, the model refers only to the latter as Polyinstantiation.

## 4.5. Multilevel Transactions

A transaction is a set of operations that read and/or write persistent objects and satisfies the *ACID* properties (atomicity, consistency, isolation, and durability). Briefly, *atomicity* means that the transaction is either executed in its entirety or not executed at all; *consistency* means that the transaction maps a database from one consistent state to another; *isolation* means that the transaction does not read intermediate results of other noncommitted transactions; and *durability* means that once a transaction is committed, its effects are guaranteed to endure despite system failures. Scheduling of transactions, i.e., locking of data, needs to be accomplished such that the user application is notified of the success or failure of each transaction. This notification, unfortunately, could lead to illegal information flows and be in conflict with confidentiality policy requirements.

We have developed the proposed model with a view toward providing an adequate foundation from which to address many issues of multilevel transactions and the ACID properties. These are introduced in the model's concepts of: Basic Object, SubObject, and Complete Object. We have identified a strategy that uses these concepts to deal with potential conflicts resulting from ACID properties. The strategy guarantees consistent transactions, and always allows adequately cleared users to make informed decisions or corrections by dealing directly with the TCB over a fully-isolated B3 trusted path.

## 5. Plans for the Future

Rome Laboratory has initiated a 30-month follow-on contract leading to development of a Trusted ONTOS proof-of-concept prototype, *TOP*, based on the work reported above. The project continues evolving the abstract access control model reported in [1] and will resolve many of the known open issues in the model.

The open issues identified to date are delineated below.

- DAC
- Auditing
- Trusted Subjects and Trusted Path
- Multilevel (Trusted) Methods
- Concurrency / Locking / Serializability
- Schema and Instance Migration
- Versioning
- Aggregates
- Backup / Recovery
- Trust Properties of Utilities

## References

[1] Schaefer, M., and S. Wade, *Requirements in Security Policy: Final Report - Preliminary Informal Access Control Model,* Contract No. MDA904-93-G-0006, CTA, Incorporated, Rockville, MD, 31 March 1994.

[2] Herndon, W., "Can We Do Without Monotonically Non-decreasing Levels in Class Hierarchies?", Unpublished Manuscript, The MITRE Corporation.

[3] Morgenstern M., "A Security Model for Multilevel Objects with Bidirectional Relationships", *Proceedings of the 4th IFIP 11.3 Working Conference in Database Security,* Halifax, England, 1990.

[4] Rosenthal, A., W. Herndon, B. Thuraisingham, and R. Graubart, "Multilevel Security for Object-Oriented Database Management Systems", Working Paper No. WP-92B0000375, The MITRE Corporation, Bedford, MA, 1993.

[5] Sandhu, R., R. Thomas, and S. Jajodia, "A Secure Kernelized Architecture for Multilevel Object-Oriented Databases", *Proceedings of the IEEE Computer Security Foundations Workshop IV,* June 1991.

[6] Sandhu, R., R. Thomas, and S. Jajodia, "Supporting Timing Channel Free Computations in Multilevel Secure Object-Oriented Databases", *Proceedings of the 5th IFIP 11.3 Working Conference in Database Security,* Shepherdstown, West Virginia, 1991.

## Discussion: Multilevel Object-Oriented Approaches

**Discussion Leader: LouAnna Notargiacomo, The MITRE Corporation**

(paper not available)

## MULTILEVEL SECURE DATABASE MANAGEMENT SYSTEM ARCHITECTURES

# Applying the Concept of TCB Subsets to Trusted Subject DBMS Architectures*

James P. O'Connor
Infosystems Technology, Inc.
1835 Alexander Bell Drive, Suite 230
Reston, VA 22091

**Abstract**

This paper presents a multilevel secure DBMS architecture that was derived by applying the concept of TCB subsets to a trusted subject DBMS architecture. The resulting architecture retains many of the advantages of a trusted subject architecture while allowing for a significantly higher level of assurance for mandatory access control. Because it is based on concept of TCB subsets, the new architecture lends itself to incremental evaluation which in turn simplifies the evaluation process, reduces the cost of re-assessing a modified system, and provides the vendor a sound basis for supporting a family of MLS DBMS products.

## 1 Introduction

The primary motivation for the development of the trusted computing base (TCB) subset approach to trusted system design was the ability to build upon a previously evaluated TCB without having to repeat any of the work that went into the evaluation of that TCB [1, 2]. This is a significant benefit, but it tends to obscure the fact that the TCB subset approach is an important general-purpose trusted system design technique. This is particularly true in the arena of multilevel secure (MLS) database management systems (DBMS) where the term "subset architecture" is often used to denote an architecture where the DBMS is completely constrained by an underlying security kernel. An alternative to a subset architecture is a "trusted subject architecture" where the DBMS contains some subjects that are not completely constrained by the underlying security kernel. In this paper, we argue that the TCB subset approach is a general-purpose design technique that can be productively applied to DBMS architectures irrespective of whether or not they employ trusted subjects.

Section 2 of this paper presents the concepts of TCB subsets and trusted subjects, and discusses the relationship between the two. Section 3 discusses MLS DBMS architectures based on each of these concepts and discusses their advantages and disadvantages. Section 4 presents an MLS DBMS architecture that combines the concepts of TCB subsets and trusted subjects, and presents the advantages and disadvantages of this architecture. Section 5 presents conclusions and future work.

*This work was supported by the U.S. Air Force, Rome Laboratories, under contract F30602-90-C-0071.

# 2 Concepts

## 2.1 TCB Subsets

In a trusted system based on the concept of TCB subsets, the overall system security policy is hierarchically partitioned and allocated to different parts (subsets) of the system. Each of these parts implements a reference monitor enforcing the corresponding policy. Each part is similar to a conventional reference monitor, with the exception that, it may use the resources of the more primitive subsets (lower in the hierarchy) to enforce its security policy (the most primitive subsets use only the hardware). A subset architecture can be incrementally evaluated, in that each of the parts can be separately evaluated against their respective policies. The evaluation of a given part depends upon the evaluation of the more primitive subsets which it uses. Even though the parts can be incrementally evaluated, it still must then be argued that when composed, parts enforce the original system security policy.

The idea of having multiple levels of security kernels, each implementing a security policy on its own objects, dates back to the design of the UCLA Virtual Machine System [3]. The current concept of TCB subsets grew out of work on the concept of extensible TCBs [4] and the first full treatment of this form of the concept was published in [1]. Here, the basic idea was to generalize the reference monitor concept [5] to support the goal of incremental evaluation of trusted systems. The Trusted Database Interpretation (TDI) [2] of the Trusted Computer System Evaluation Criteria (TCSEC) [6] embraced the concept of *hierarchically* related subsets as a basis for trusted DBMS development and evaluation.

The TDI formally defines a subset M as the a set of software, firmware, and hardware (where any of these three could be absent) that mediates the access of a set S of subjects to a set O of objects on the basis of a stated access control policy P and satisfies the properties:

1. M mediates every access to objects in O by subjects in S;
2. M is tamper resistant; and
3. M is small enough to be subject to analysis and tests, the completeness of which can be assured.

Furthermore, the TDI specifies a set of conditions that a subset architecture must meet in order to be eligible for an evaluation by parts. These conditions are:

1. The candidate TCB subsets are identified;
2. The system policy is allocated to the candidate TCB subsets;
3. Each candidate TCB subset M[i] includes all the trusted subjects with respect to its technical policies P[i];
4. The TCB subset structure is explicitly described;
5. Each TCB subset occupies distinct subset-domains; and

6. The more primitive TCB subsets provide support for the reference validation mechanism arguments for the less primitive TCB subsets.

Any architecture that claims to be a subset architecture must satisfy these conditions.

## 2.2 Trusted Subjects

A trusted subject is an entity (usually a process) that runs with special privilege that allows it to bypass the security policy of an underlying reference monitor[1]. For example, UNIX System V Release 4.2 ES allows a subject to be granted a number of privileges, including: read-up (MACREAD), write-down (MACWRITE), and modify process level (SETPLEVEL) [7]. Other operating systems provide a more granular privilege mechanism where processes are only trusted within a range [8].

Trusted subjects are employed in a system design when the constraints implemented by the underlying security mechanism make it impossible (or very difficult) to implement required functionality. They may be used as part of the implementation of a multilevel secure operating system (and hence are evaluated as part of the operating system evaluation), or they may be added later to support a trusted application (e.g., a guard application). Since trusted subjects are not completely constrained by the underlying reference monitor, it is crucial that they be carefully analyzed to ensure that they do not violate the intended security policy.

## 2.3 Relationship Between Subsets and Trusted Subjects

The first issue that must be addressed is whether it is possible to apply the concept of TCB subsets to an architecture utilizing trusted subjects. One way to illustrate that the concepts of trusted subjects and TCB subsets are compatible, is to start with a valid TCB subset architecture, add a trusted subject, and argue that the result can be made into a valid (but different) subset architecture. A subset architecture is valid if it satisfies the six criteria for a subset architecture, and each subset possesses the three reference monitor properties required of a subset.

Assume that there exists a TCB that is layered into n hierarchical subsets M[0], M[1], ..., M[k], ..., M[n], and is valid by the above definition. Now suppose that we add a trusted subject to subset M[k] (trusted with respect to subset M[k-1]). This situation is shown on the left side of Figure 1. As modified, this architecture is not valid because it violates condition 3. However, we can combine the subsets M[k] and M[k-1] into a single subset that is allocated the combined policies of the two subsets.[2] The resulting architecture, which is shown on the right side of Figure 1, is a new candidate subset architecture.

It should be apparent that the candidate architecture satisfies the six conditions for a subset architecture as listed above. The remainder of this section argues that the new subset (M[k-1]) can satisfy the required reference monitor properties as well. The first property

[1]Even though most discussions of trusted subjects focus on the ability to circumvent mandatory access control, a process can be trusted with respect to any aspect of the policy implemented by the reference monitor (e.g., discretionary access control).

[2]In general, a trusted subject introduced in subset M[i], that is trusted with respect to level M[j], $i > j$, will require all subsets M[k], $i \geq k \geq j$, to be combined.

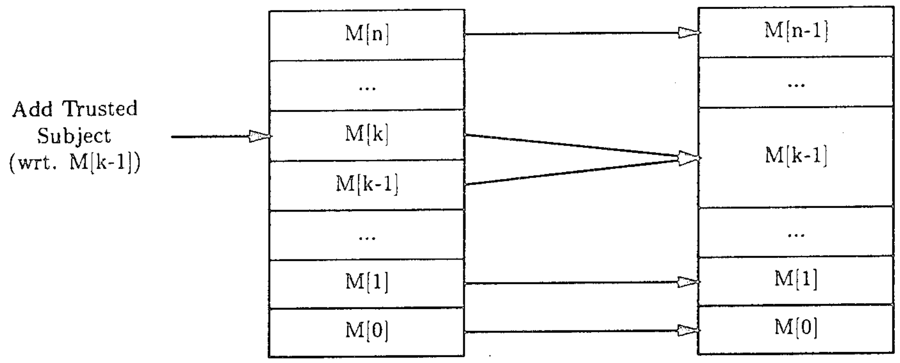


Figure 1: Adding a Trusted Subject to a Subset Architecture.

can always be satisfied because the new subset can use the same mechanisms as the old M[k] and M[k-1] subsets to ensure that it is not bypassed. The second property can always be satisfied because the new subset can use the same mechanisms as the old M[k] and M[k-1] subsets to ensure that it is tamper resistant. The satisfaction of third property depends on characteristics of the original subsets that were combined. As noted in the TCSEC, the third property is currently interpreted to mean that the TCB "must be of sufficiently simple organization and complexity to be subjected to analysis and tests, the completeness of which can be assured" [6]. It certainly can be argued that if the original two subsets satisfied this criterion, and the additional trusted subject satisfied this criterion, then the new subset would satisfy it. At higher assurance levels there is also the implication that modules that are not protection critical have been excluded from the TCB. A similar argument could be made that if the original two subsets were in some sense "minimal" given their respective polices, the combined subset would also be minimal for the combined policy, provided that the trusted subject itself were minimal.

The above argument demonstrates that, although trusted subjects have a definite impact on subset architectures, they can not be determined a priori to be incompatible concepts. The implication is that an MLS DBMS can be implemented using trusted subjects and may still derive benefit from an application of the concept of TCB subsets.

## 3 MLS DBMS Architectures

### 3.1 TCB Subset DBMS Architecture

The concept of TCB subsets has been applied in the domain of database architectures to produce a TCB subset DBMS architecture. This architecture was first proposed as part of the SeaView effort [9]. In this architecture, the DBMS runs as one or more untrusted processes on top of a security kernel. This architecture consists of two subsets. The most primitive subset is the underlying security kernel, which is responsible for all mandatory access control enforcement. The DBMS forms a second subset which enforces a discretionary access control (DAC) policy on its own objects.

The advantages of this architecture are ease of evaluation and assurance. The ease of evaluation of this architecture is due to the fact that, since this architecture has no trusted subjects, the DBMS is prevented from doing anything that would invalidate a previous evaluation of the underlying security kernel. There is no need to perform any re-evaluation of the underlying operating system. The mandatory assurance characteristics of this architecture are derived directly from the mandatory assurance characteristics of the underlying operating system. For this reason, the mandatory assurance level of the DBMS should be the same as that for the underlying security kernel.

The disadvantages of this architecture are that it is inflexible, difficult to use, difficult to implement, and inefficient. The architecture is inflexible because:

1. *Polyinstantiation is unavoidable.* This is because the presence of similarly named objects at higher or non-comparable levels cannot be detected. This is true not only of tuples, but of databases, relations, and schemata.

2. *Integrity constraints cannot always be enforced.* This is because the enforcement of certain constraints require the ability to detect the presence of objects at a higher or non-comparable level or to remove objects at a lower or non-comparable level.[3]

3. *DBMS Trusted subjects cannot be supported.* This is because the DBMS itself cannot circumvent MAC privileges, therefore it cannot offer any such services to its clients.

4. *Information cannot be downgraded.* This is because downgrading requires a trusted subject, which is not permitted in a TCB subset DBMS architecture.

The resulting DBMS is difficult to use for the above reasons as well as the fact the database dumps, restores, and bulk loads must be performed at each level in the security lattice. The implementation is difficult because:

1. *Data must be fragmented.* This is because the DBMS must store all multilevel data, metadata, and log information in the single level objects provided by the security kernel.

2. *Concurrency control and recovery must be performed without global knowledge.* Since the subjects that perform these operations are necessarily single level, they can only see the portion of the relevant data that they dominate.

3. *DBMS processes must be replicated at each security level.* Since the DBMS subjects are necessarily single level, there must be one for each client security level to be supported.

Finally, the architecture is less efficient because:

1. *The amount of I/O is increased because of data/log fragmentation.* This is because the DBMS subject must read from one file for each level in the security level lattice that it dominates. This will significantly reduce the effectiveness of buffering.

[3]It can be argued that this and the previous "disadvantage" are actually necessary characteristics of a secure system (because both failure to support polyinstantiation and complete enforcement of integrity constraints can introduce covert channels). The issue is that a system based on this architecture cannot give the DBA the option to trade-off security and data integrity.

2. *The DBMS must rely on the operating system file management.* Database management systems frequently implement their own file systems that are optimized for database access. Since these file systems are necessarily multilevel, they cannot be implemented using untrusted subjects.

3. *Each DBMS process must be duplicated at each security level.* As noted above, the DBMS TCB subset architecture requires the duplication of processes by security level. This will have a negative impact on performance because of the additional context switching overhead (and resource consumption).

### 3.2 Trusted Subject DBMS Architecture

In the trusted subject DBMS architecture, the DBMS includes one or more subjects that are trusted with respect to the security policy of the underlying operating system. The composed system (operating system plus application) implements a security policy that is potentially different from the one originally implemented by the operating system.

The advantages of this architecture are performance, flexibility, ease of implementation, and ease of use. The primary disadvantages of this architecture are low assurance and evaluation difficulty. Both of these disadvantages are a result of the fact that, since the DBMS is not fully constrained by the underlying operating system TCB, flaws in its implementation can cause a breach of mandatory security. The difficulty of evaluating such a system is compounded by that fact that the combination of the trusted subject and the TCB of the underlying operating system can introduce information flows that cannot be discovered by performing an analysis of the trusted subject alone. The implication of this is that it is not sufficient to look at the trusted subject alone when evaluating the security characteristics of the DBMS. At least some of the evaluation of the underlying operating system must be repeated.

## 4 Proposed Architecture

The trusted subject and TCB subset architectures presented in the previous section are generally considered to be disjoint, each having its own distinct advantages and disadvantages [10]. This view is not consistent with the idea of TCB subsets as a general-purpose design technique. As discussed in section 2.3, there is no technical reason why the concept of TCB subsets cannot be productively applied within the domain of trusted subject DBMS architectures. This section presents an MLS DBMS architecture derived by applying the concept of TCB subsets to a trusted subject DBMS architecture.

### 4.1 Architecture Definition

Figure 2 shows an abstract MLS DBMS architecture that was derived by applying the concept of TCB subsets to a trusted subject DBMS architecture. This architecture consists of two subsets, M[0] and M[1]. The M[0] subset enforces a mandatory access control policy on DBMS objects (e.g., tuples) and consists of the operating system TCB combined with the minimal amount of trusted DBMS code required to implement the desired policy. The M[1] TCB subset is layered upon M[0] and enforces a discretionary access control policy

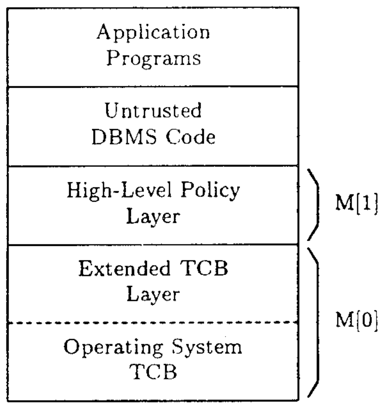


Figure 2: Abstract Subset Architecture.

that is a refinement of the policy enforced by the M[0] TCB. Each of these subsets must be isolated via a domain isolation mechanism (e.g., protection rings [11]).

This architecture is abstract in the sense that it can describe a wide range of actual MLS DBMS systems. One important detail that has been omitted is the actual allocation of DBMS functionality to the subsets. For the M[0] subset, some possibilities include: no DBMS functionality (the conventional TCB subset case), a simple filter, and significant DBMS functionality (e.g., access methods and scheduler). For the M[1] subset, some possibilities include: a null subset (i.e., rely on the DAC, if any, provided at the M[0] subset) and subsets whose functionality is determined by the granularity of the definition of protected objects in the system's security policy (e.g., relations, columns, views).

The M[0] subset of this architecture actually consists of two "parts" (in the TDI sense) that are isolated in separate protection domains. Since one of these parts contains subjects trusted with respect to the other, these two parts do not qualify for an evaluation by parts. As noted in the TDI, even though these parts do not qualify for an evaluation by parts, it is likely that significant savings can be recognized by reusing the results of the evaluation of the underlying security kernel. The problem encountered here is that there is no theory that can be used to quantify these savings a priori.

It is also worth noting that the M[1] subset (or the M[0] subset for that matter) could be further subdivided into additional subsets if desired. This would of course depend on a meaningful decomposition of the security policy and the availability of the required domain isolation mechanisms.

The remainder of the paper will focus on an instance of the above architecture in which the extended TCB layer includes the *minimal* DBMS functionality required to retain the significant advantages of the trusted subject architecture as discussed in 3.2.

## 4.2 Advantages

The proposed architecture retains the significant advantages of the trusted subject DBMS architecture while mitigating its disadvantages. The advantages are retained through the judicious use of trusted subjects (e.g., to avoid data fragmentation). The disadvantages are mitigated by isolating all DBMS code that requires mandatory privilege to the lowest level subset. The result is a system that offers significantly higher assurance for mandatory access control and is more evaluatable than a similar system with a monolithic TCB. The assurance advantages are a result of the fact that:

- *The amount of code that can cause a violation of MAC is significantly decreased.* The amount of DBMS code in the M[0] subset is significantly less than that in the TCB as a whole *and* only subjects in this subset can run with special MAC privileges. Since subsets must satisfy the isolation and non-bypassability requirements for a reference validation mechanism, these properties guarantee that only code in the M[0] subset can cause a violation of MAC.

- *The effectiveness of assurance techniques is increased.* Assurance techniques are more effectively applied at a lower level of abstraction. Since assurance techniques must be applied to each subset, the TCB subsets approach forces you to apply these techniques more directly to the portions of the system responsible for MAC enforcement (viz., the M[0] subset). Additionally, if you subscribe to the notion of balanced assurance [12], this approach has the effect of focusing your assurance efforts where they will have the most impact.

The proposed architecture is easier to evaluate because:

- *The scope of global analysis is reduced.* Developing a system using trusted subjects requires that certain global analysis be performed on the combined underlying TCB and the DBMS TCB (e.g., covert channel analysis). If the DBMS TCB has a multi-subset TCB, only the M[0] TCB must be considered in these global analyses.

- *The evaluation task can be partitioned.* One of the primary benefits of the TCB subsets approach is the ability to divide a complex system into parts and evaluate the parts incrementally. This approach makes the evaluation of a complex TCB more tractable.

- *The re-assessment of modified or ported systems is simplified.* A TCB subset architecture has the characteristic that the evaluation impact of certain changes is isolated to the subset in which they occur. This can result in significant savings in the area of re-assessment.

- *Subsets can be evaluated to different assurance levels.* This architecture has the characteristic that the different subsets can be evaluated to different assurance levels. That is, the M[0] subset could be evaluated to a relatively high level (e.g., B3 or A1) while the M[1] subset could be evaluated at a lower level (e.g., C2).[4]

In addition to the assurance and evaluation benefits, applying the concept of TCB subsets to a trusted subject architecture permits a vendor to support a family of MLS DBMS products without duplicating evaluation effort. A vendor could support an entire product line (e.g., with products supporting different DAC policies) with the basic M[0] TCB at its core. Since a subset architecture allows incremental evaluation, the underlying M[0] TCB need only be evaluated once for the entire product line.

[4]Current evaluation practice is to require all subsets to be evaluated at a uniform assurance level. There is, however, no technical reason to require a uniform assurance level provided that a given subset does not depend upon a less assured subset.

### 4.3 Disadvantages

The application of the concept of TCB subsets to trusted subject DBMS architectures has some disadvantages as well. Specifically, the proposed architecture has the following disadvantages:

- *Implementation difficulty associated with multiple protection domains.* The architecture presented above will require at least four hierarchical protection domains: one for the operating system, one for the extended TCB layer, one for the M[1] subset, and one to protect the integrity of the untrusted DBMS code. These domains can be provided through a variety of mechanisms and each domain need not use the same mechanism.
- *Performance overhead associated with multiple protection domains.* As noted above, this architecture requires at least four hierarchical protection domains. Crossing domain boundaries is likely to have a negative impact on DBMS performance.
- *Reduced Flexibility.* This reduction in flexibility occurs because the M[1] subset cannot violate policy of the M[0] subset even in cases where it would be desirable. For example, it would not be possible to support trusted stored SQL procedures in a DBMS in which SQL is outside of the mandatory subset. Exactly how much flexibility is lost is determined by what DBMS functionality is placed in the M[0] subset.

These disadvantages are significantly less that those realized in the conventional TCB subset DBMS architecture.

## 5 Conclusions

This paper proposed an MLS DBMS architecture that was derived by applying the concept of TCB subsets to a trusted subject DBMS architecture. The proposed architecture retains the strengths of the trusted subject architecture while mitigating its weaknesses. The strengths of the trusted subject architecture are its performance, flexibility, ease of use, and ease of implementation. These strengths are retained in the proposed architecture through the judicious use of trusted subjects. The weaknesses of the trusted subject architecture are mandatory assurance and evaluation difficulty. A significantly higher level of mandatory assurance is possible in the proposed architecture because the amount of code that requires mandatory privilege is minimized and that code is isolated in the lowest level subset. The evaluatability of the architecture is improved because it can be evaluated incrementally, and certain global analyses are only required on the lowest level subset. In addition, an incremental evaluation can reduce the cost of re-assessing a modified system, and provide the vendor a sound basis for supporting a family of MLS DBMS products.

We are currently prototyping the proposed architecture by reengineering the Trusted RUBIX MLS DBMS [13]. The anticipated results of the prototyping effort are a demonstration of the feasibility of the new architecture and an improved understanding of the properties of the architecture. Topics to be investigated include: size of the extended TCB, trade-offs between extended TCB size and architectural flexibility, performance characteristics, and resource utilization characteristics.

## References

[1] W. Shockley and R. R. Schell. TCB subsets for incremental evaluation. *Proceedings of the Third Aerospace Computer Security Conference*, December 1987.

[2] National Computer Security Center. Trusted database interpretation of the trusted computer system evaluation criteria. Technical Report NCSC-TG-021, National Computer Security Center, April 1991.

[3] G. J. Popek and C. S. Kline. A verifiable protection system. *Proceedings of the International Conference on Reliable Software*, pages 294–304, 1975.

[4] M. Schaefer and R. R. Schell. Toward an understanding of extensible architectures for evaluated trusted computer system products. *Proceedings of the 1984 Symposium on Security and Privacy*, pages 41–49, 1984.

[5] J. P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51 (AD-758206), J. P. Anderson Co., October 1972.

[6] Department of Defense. Department of Defense trusted computer system evaluation criteria. DOD Standard 5200.28-STD, Department of Defense, December 1985.

[7] Unix System Laboratories. *System V Release 4.1 ES Network User's and Administrator's Guide*, 1991.

[8] R. R. Schell, T. F. Tao, and M. Heckman. Designing the Gemsos security kernel for security and performance. *Proceedings of the 8th National Computer Security Conference*, 1985.

[9] Teresa F. Lunt and Peter K. Boucher. The SeaView prototype: project summary. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, Maryland, October 1994.

[10] W. Timothy Polk and Lawrence E. Bassham III. Security issues in the database language SQL. NIST Special Publication 800-8, National Institute of Standards and Technology, August 1993.

[11] M. D. Schroeder and J. H. Saltzer. A hardware architecture for implementing protection rings. *Communications of the ACM*, 15(3):157–170, March 1972.

[12] T. Lunt, D. Denning, R. Schell, M. Heckman, and W. Shockley. Element-level classification with A1 assurance. *Computers and Society*, August 1988.

[13] J. P. O'Connor. Trusted RUBIX: A multilevel secure client-server DBMS. In *Proceedings of the Eigth Annual IFIP Working Group 11.3 Working Conference on Database Security*, August 1994.

# SWORD on CMW

Simon R. Wiseman

Defence Research Agency
St. Andrews Road
Malvern, Worcestershire, WR14 3PS, England

wiseman@dra.hmg.gb

## Abstract

CMWs offer floating labels as a means of providing flexible secure operation. The SWORD secure DBMS provides fine grain protection at the field level, with labels that do not float. SWORD also supports clients which are trusted to label their queries appropriately. By judiciously making the CMW labels float according to the result of a query, an ordinary CMW process can be allowed the flexibility of a trusted client. This preserves some of the advantages of floating labels without complicating the semantics of the DBMS.

## 1. Introduction

The SWORD secure DBMS [1] was designed to overcome the problems that had been perceived with the use of polyinstantiating DBMSs [2]. It was intended to be used as part of systems operating in multi-level security mode, with labelled entities forming the basis of information flow security. However, Compartmented Mode Workstations (CMWs) [3] have since become an important component of secure systems and these, unlike other secure components, use a dual-labelling system.

When DBMSs, or other secure subsystems, interact with CMWs, consideration must be given as to how the DBMS's simple single-label scheme interacts with the dual-labels in the CMW. However, in order to judge how this may be done effectively, it is necessary to ascertain what role the CMW dual-labels are intended to fulfil.

The important feature of CMW dual-labels is that they provide floating labels. These float in response to the flow of labelled data and offer a distinct advantage with respect to operating Commercial-Off-The-Shelf (COTS) software in a secure system. In an ordinary secure Unix, COTS software would be constrained to work at a single security level. If the software needed to read files classified higher, or write files classified lower, than its "session level" it would be prevented from doing so by the secure operating system. Since the software is unaware of labels, it would not be expecting its read/write requests to be rejected and will at best provide meaningless error messages to the user.

With floating labels, COTS software is not constrained by the operating system's labels, even though it is unaware of them. If the software reads a highly classified file, the process' label is raised. If the software writes to a lowly classified file, the file's label is raised. Problems occur only if the software tries to work above the user's clearance, but this would not be expected in normal operation.

This paper briefly discusses how CMW dual-labels may be used and describes SWORD's security controls. Then, one way in which SWORD may be used in a CMW environment is described.

## 2. CMW Dual-Labels

A CMW applies two labels to each entity: a Sensitivity Label and an Information Label. The Information Label is initially set to lattice-bottom (the lowest possible label), but it floats upwards as more classified data is moved into the entity. The Sensitivity Label is generally fixed and acts as an upper bound to the Information Label.

The Information Label of an entity in a CMW is often considered to be a more appropriate reflection of the sensitivity of the information contained in the entity than the Sensitivity Label. However, the exact difference between the Information Label and Sensitivity Label is not precisely specified. This is unfortunate since it is then difficult to decide whether some use of the Information Label is appropriate.

It would seem reasonable to assume that, for active entities which communicate directly with the user (such as windows), the user's clearance is used as the Sensitivity Label[1]. This reflects the fact that the sensitivity of information presented to the user should not be higher than their clearance. Sensitivity Labels on other entities, which are internal to the machine (such as files), are also needed as part of the mechanism to ensure that Information Labels on windows do not float too high [4].

The Information Label is intended to provide the classification which should be applied to the data within an entity. However, classified information may theoretically be encoded in the state of an entity, without this being reflected in the entity's Information Label. This is because the Information Label only accounts for information flows that occur when data moves [4].

The CMW design allows the Information Label to ignore further subtle information flows, which arise because of the way entities are addressed and printed outputs are labelled [4]. Also, some implementations of CMW ignore various other flows relating to addressing.

It is hard to conceive of a system where meaningful information flows when data ceases to move, unless the system is being actively attacked by sophisticated opponents using Trojan Horse techniques. In low threat environments, vulnerabilities that can only be exploited by Trojan Horses are considered to be an acceptable risk[2]. The additional flows allowed by the design, and introduced by the implementations, also seem to be difficult to exploit. Thus, the Information Label is likely to suffice as an accurate indication of the protection required for an entity's contents.

To summarise, it would appear that there are two possible ways of using the dual labels of a CMW.

1. The Information Label is just some additional data about an entity, which is probably used during downgrading operations to convey the requested new security class.
   The Sensitivity Label gives the protection required of the data within an entity.

2. The Information Label gives the protection required of the data within an entity.
   The Sensitivity Label is just an upper bound that is used to carry the user's clearance.

---

[1]Though the Sensitivity Label might be artificially lower than the users true clearance.
[2]The terminology of [11] is used, so roughly *risk = vulnerability * threat*

In the first case, the Information Label is relatively uninteresting. Its value is not really relevant to the security of the overall system, because it plays no part in the enforcement of confidentiality. If it becomes set to an inappropriate security class, the user who is to review the requested downgrade will reject the request.

In the second case, the Sensitivity Label is relatively uninteresting, since it just conveys the user's clearance. It is only needed in those systems where not all users have the same clearance, or where some workstations cannot be used to access all information because they are in a less-well protected area.

In the first case, the way in which the Information Label is affected by interaction with a secure DBMS is not particularly crucial. However, in the second case it is critical. Thus, it is important to consider the second case in deciding how a DBMS and CMW should interact.

## 3. SWORD Security Controls

SWORD is a secure Relational DBMS which provides field level labelling, without forcing designers to Polyinstantiate [5] or circumvent information flow security by using privileged clients.

The field labelling in SWORD is not equivalent to row labelling, unlike the field labelling schemes of SeaView [6]. In SWORD the existence of a highly classified field is generally classified low, while in SeaView the existence of a field is always classified the same as its contents. This means that SWORD only allows rows to be inserted by clients with low clearances - the Insert Low approach [7]. A consequence of using this approach to support multi-level databases, is that clients are able to attempt to observe the contents of a field for which their clearance is insufficient. In SWORD the result of such attempts is a special "not cleared" value [8].

SWORD has been designed so that a database can be maintained and operated by untrusted clients. However, it does also support clients that can be trusted to label queries appropriately [9]. A trusted client is not, however, trusted to avoid queries that cause inappropriate downward information flows within the database. In SWORD, even trusted clients are prevented from causing a downward flow in the databases. This constraint is enforced because the effect of a query depends greatly upon the data in the database, and so it is difficult to have confidence that the effect of a general downgrading query is always limited to affecting the data envisaged.

Thus, SWORD does not support downgrading of data *in situ*, so downgrading must be performed in the application. This is not thought to be unreasonable, since downgrading is generally subjected to stringent application specific controls, which can only be carried out in the application.

A trusted client of SWORD is free to indicate the sensitivity of information encoded in the text of a query and the fact that the query is issued, even if this is strictly lower than the clearance with which it is to be evaluated[1]. The advantage of doing this is that the fact that changes are occurring is often less sensitive than exactly what is changing. For example, the insertion of a new row into a table may be much less sensitive than the values placed in some of the fields.

SWORD also provides detailed information labels on the results of select queries, which indicate the source of the information it conveys [10]. SWORD's information labels are provided on both

[1]For untrusted clients, the sensitivity of the query (its text and the fact that it was issued) must equal the clearance.

the fields and rows of the result. An information label in SWORD is not a floating label, unlike the Information Labels in CMWs. In SWORD, the information label states the minimum clearance required to ascertain some basic facts, specifically ignoring the reason that the basic facts were retrieved.

The information label of a row is given by the least clearance required to ascertain that the select query's where-clause expression is true for that row. This is discounting the fact that the text of the where-clause may itself be sensitive information. The information label of a field is given by the least clearance required to ascertain the value of the select list expression, in the corresponding position, for that row. This is ignoring the fact that the row was selected, which may itself be based on sensitive information.

In effect, the information labels state the sensitivity of the result, excluding sensitive information derived from the addressing information which caused them to exist, ie. the where-clause and select list expressions. They indicate which users with lower clearances may learn the same basic facts, even if they must issue different queries to do so.

## 4. CMW Active Entities as SWORD Clients

An active entity in a CMW may become a client of a SWORD database by connecting to it. From SWORD's point of view the client must have a clearance, which is the maximum sensitivity of information that will be returned to it. The Sensitivity Label of the client acts as an upper bound on the sensitivity of information which may be included in the entity, thus the SWORD client's Clearance is obviously the CMW entity's Sensitivity Label.

A query's text, and the fact that it is issued, is derived from the contents of the active entity. Thus, the sensitivity of this information equals the sensitivity of the active entity's contents. Hence, it seems reasonable that the query should be labelled with the active entity's Information Label.

The results of a query may be computed from information of a strictly higher sensitivity compared with the active entity's Information Label. In this case it might seem reasonable to float the Information Label to reflect this information flow. However, the CMW Information Labels are not completely accurate, in that they ignore certain information flows such as those relating to addressing. Thus, it might be appropriate to ignore some of the flows that occurred during query evaluation.

In particular, a select query generally does not bring back a result for every row, usually because the where-clause expression yields false for some rows. The fact that a row is not selected is actually a flow of information back to the client, which reveals something about the values in the ignored row. It might be reasonable to ignore this flow on the grounds that it is rather covert, requiring the text of the query and the result to be tied together with details about what was not retrieved in order to obtain "useful" information.

A similar argument could be made for ignoring the fact that the where-clauses of rows that are selected all evaluate to true. However, this information is slightly more obvious since it does not require additional knowledge.

In effect, ordinary untrusted CMW application software acts like a trusted client from SWORD's point of view - assuming that the risk of using the Information Labels to protect information in the way described is acceptable, given the perceived threat.

Genuinely trusted CMW software would be permitted to bypass the controls imposed by the dual-labelling. For example, CMW software that is trusted to lower its Information Label would also be permitted to set the label of a query strictly lower than the Information Label.

## 5. The SWORD Prototype

The SWORD prototype runs on Sun CMW and is implemented by front-ending standard Ingres. The front end floats the active entity's Information Label by the information labels of all the rows and fields in the result. Thus it discounts the flows arising from computing the where-clauses of rows that are not returned. It also ignores the flows arising from update and delete queries, which report back the number of rows affected. Strictly, this number reveals something about the fields observed during the evaluation of the where-clause expression.

The following example shows the effect. The information label of the resulting row is Confidential, since a clearance of Confidential is required to compute the where-clause for that row.

Flights table:

| Name | | Dest | | Mission | |
|---|---|---|---|---|---|
| "Enterprise" | [U] | "Vulcan" | [U] | "Supply" | [C] |
| "Constitution" | [U] | "Romulus" | [C] | "Attack" | [S] |

| | |
|---|---|
| Query: | SELECT Name FROM Flights WHERE Mission <> "Attack"; |
| Query Sensitivity: | Unclassified (CMW Information Label) |
| Clearance: | Secret (CMW Sensitivity Label) |

Result:

| | Name | |
|---|---|---|
| [C] | "Enterprise" | [U] |

Resulting CMW Information Label: Confidential ([U] lub [C] lub [U])

From the result and the query, it is possible to deduce the Confidential information that the Enterprise is not on an Attack mission. This is reflected in the resulting Information Label. However, using the query "SELECT name FROM Flights;" to ascertain that no row for the Constitution has been selected, it is also possible to infer that Constitution is on an Attack mission. This information is derived from a Secret field, but this is not reflected in the Information Label.

Thus SWORD used on a CMW in this way introduces additional vulnerabilities into the use of Information Labels to protect classified information. However, these appear to be commensurate with existing vulnerabilities that arise due to the controls over addressing entities, and it is expected that in many systems the perceived threat is low enough to make the risk acceptable.

## 6. Conclusions

The dual-labelling system of CMWs can be used in a number of ways. The most useful appears to be to directly use the floating Information Label to protect classified information. This method allows label unaware COTS software to work unhindered in the presence of multi-level security functionality.

The Information Label in a CMW does not account for all information flows. Thus there is a question as to what information flows from a secure DBMS should be taken into account. Unfortunately, this is an application specific decision which can only be made on the basis of a security risk assessment. Hence, secure DBMSs must be prepared to interact with CMW dual labels in a variety of ways.

The main question is how the client's Information Label should float in response to the results of queries. The most obviously secure way is to float it up to the Sensitivity Label, but this seems rather strong. The SWORD front end prototype is experimenting with another possibility, where certain subtle information flows, that arise because of the way data is addressed, are ignored. Other alternatives exist, for example the label could float according to the sensitivity of all fields examined during query evaluation, however these would appear to be less effective, although stronger.

SWORD does not provide floating labels in the database, and therefore does constrain COTS software to an extent. For example, if a client attempts to update a lowly classified field with a highly classified query (high Information Label), the request will fail - the field label does not float. It remains to be seen whether this will cause practical problems, or whether floating labels in a database are a practical proposition.

## 7. References

[1] The SWORD Multilevel Secure DBMS
A.W.Wood, S.R.Lewis & S.R.Wiseman
RSRE Report 92005, February 1992

[2] On the Problem of Security in Data Bases
S.R.Wiseman
Proceedings 3rd IFIP WG11.3 Working Conference on Database Security
Monterey, CA, September 1989

[3] Compartmented Mode Workstation Evaluation Criteria, Version 1 (final)
Defense Intelligence Agency report DDS-2600-6243-91, November 1991

[4] Using Security Models to Investigate CMW Design and Implementation
Clare L. Robinson & Simon R. Wiseman
To be presented at the 1994 Computer Security Applications Conference
Orlando, FL, December 1994

[5] Notes on the Polyinstantiation Problem
S.R.Wiseman
RSRE Memorandum 4504, July 1991

[6] Tuple-level vs. element-level classification
Xiaolei Qian & Teresa Lunt
Proceedings 6th IFIP WG11.3 Working Conference on Database Security
Vancouver, BC, August 1992

[7] Control of Confidentiality in Databases
S.R.Wiseman
Computers and Security Journal, Vol 9, Num 6, pp529-537, October 1990

* All RSRE reports, RSRE memoranda and DRA reports are available from the author.

[8] Extending SQL to Support Secure Applications
S.R.Wiseman
DRA Report DRA/CIS/CSE2/TR94001, July 1994

[9] Field Level Classification and SQL
Simon R. Wiseman
To be presented: 8th IFIP WG11.3 Working Conference on Database Security
Bad Salzdefurth, Germany, August 1994

[10] Security Properties of the SWORD Secure DBMS Design
Simon R. Wiseman
Proceedings 7th IFIP WG11.3 Working Conference on Database Security
Huntsville, AL, September 1993

[11] Glossary of Computer Security Terms
CESG Computer Security Memorandum No. 1, issue 2.2
November 1993

# LOCK DBMS: Integrating Type Enforcement

Dan Thomsen, Dick O'Brien and Tom Haigh

Secure Computing Corporation
2675 Long Lake Road
Roseville, Minnesota 55113

## Introduction

SCC's LOCK DBMS program is developing an Exploratory Development Model (EDM) of a high assurance, state-of-the-art Trusted Database Management System (TDBMS) on the LOCK/SNS platform. The SNS system is a highly assured platform that is based on the LOCK prototype, which was designed to meet the Class A1 requirements of the Trusted Computer Security Evaluation Criteria (TCSEC) [1]. The commercial TDBMS that is being used for LOCK DBMS is Trusted Oracle Version 7 [2]. The LOCK DBMS EDM is scheduled for demonstration in June, 1994.

The LOCK DBMS EDM uses a TCB subset architecture to allow the underlying SNS system to enforce a high assurance Mandatory Access Control (MAC) policy [3], [4]. One of the original goals of the LOCK DBMS program was to investigate how the LOCK/SNS Type Enforcement mechanism could be used to provide additional integrity and security to a commercial-off-the-shelf (COTS) TDBMS. The areas of particular interest were:

1 providing strong separation between the database entities (files and processes) and other system entities
2 providing high integrity auditing on database objects
3 providing high integrity DAC enforcement on database objects
4 integrating high integrity SNS roles with DBMS roles.

This paper reports on the results of these efforts and identifies areas where future research is needed.

## Incorporating Type Enforcement into the EDM

The LOCK/SNS Type Enforcement mechanism [5] is used to restrict the access of subjects (processes) to objects (data) and other subjects. A *type* is associated with each object and a *domain* with each subject on the system. The access a subject is permitted to an object depends on the access capability that the subject's domain is permitted to the object's type. Furthermore, the access a subject is permitted to another subject depends on the access capability that the first subject's domain is permitted to the second subject's domain.

In the LOCK DBMS EDM using special database domains and types, the database system is completely isolated from the rest of the system, and access to the database files is restricted, in a mandatory manner, to the TDBMS subjects. This prevents accidental or malicious access to the database by other system subjects and shows that objective 1 in the above list can be successfully achieved.

Without redesigning the Oracle TDBMS, however, the remaining objectives in the list, can only be achieved either partially or not at all This fact is a consequence of the underlying Oracle architecture. As Figure 1 shows, all Oracle database processes, including the servers, have access to the Shared Global Area (SGA) object. Since the servers can communicate through this shared object, Type Enforcement cannot be used to ensure that separate server subjects do not share information. To achieve such separation would require that each server have a separate SGA, which implies that a separate instance, with all of its overhead, would be needed for each server. While this approach has been used on the LOCK DBMS EDM to provide special mandatory roles, in general, it is not appropriate to provide a stronger DAC because of the negative performance and administrative impact.

In the following sections some other possible approaches and research issues are discussed relating to objectives 2, 3 and 4.

## Providing Audit on Database Objects

In an MLS database, auditing must be performed at the granularity of database objects and, hence, is done by the TDBMS. In Trusted Oracle, DBMS auditing is done by recording SQL statements as they are received. The assurance of the DBMS audit is not high since Trusted Oracle has only a B1 level of assurance. A possible approach

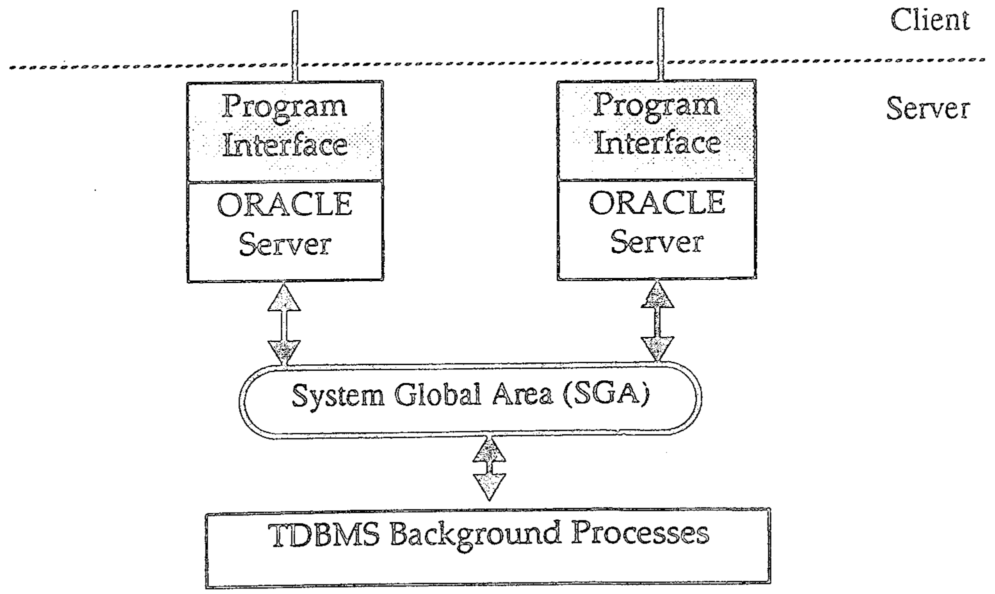


Figure 1 An ORACLE Instance.

*All processes in the instance, including the servers, share a common global area (the SGA object) through which information can flow. The architecture prevents Type Enforcement from being used to separate servers, acting for different users in different domains. The only current solution is have separate instances for each server so that they do not share the same SGA.*

that would increase the assurance of this type of auditing would be to monitor the communication channel between the client and the DBMS server. Type Enforcement can be used to ensure that all user requests to the server must go through a request auditor, as illustrated in Figure 2. The request auditor would be a small piece of code that logs all user SQL requests to the LOCK audit trail.

The drawback to this approach is that there is no higher assurance that the information returned to the user or the operation performed is what the user requested and, hence, what the audit record shows. Since the assurance of the server is not increased, it could still perform any operation in response to the user's query.

## Enforcing DAC on Database Objects

As mentioned earlier, a higher assurance DAC could be achieved by running separate instances for each user, but this approach is not realistic. A less drastic approach would be to partition the SGA into separate objects that could be protected by Type Enforcement or the higher assurance DAC on files provided by the underlying TCB. This approach is not really feasible, however, since it prevents DAC from being enforced with the flexibility and granularity that DBMSs use, and it also sacrifices the performance gain that is achieved by sharing data and SQL code (e.g. shared stored procedures).

The only other alternative would be to provide higher assurance to those components of the TDBMS that provide the DAC enforcement. Such an approach will probably entail redesigning the TDBMS to separate the DAC component from the remainder of the system. How to implement high assurance DBMS DAC, and whether it is even necessary, remains an open research question.

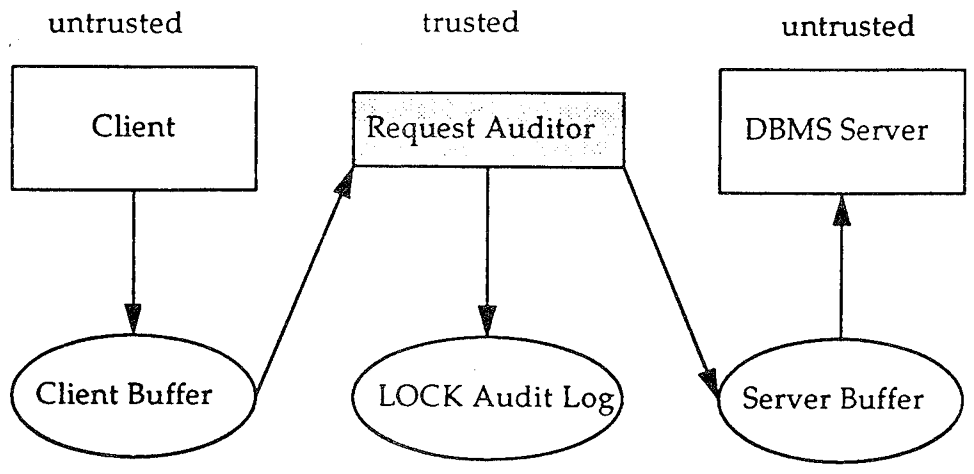


**Figure 2 High Integrity User Request Auditing**

*All user requests to the database must go through the request monitor which is trusted to log the request to the LOCK audit log.*

## Enforcing Roles

Both SNS and Trusted Oracle have their own concept of roles. The SNS Type Enforcement mechanism allows roles to be implemented in a mandatory manner with A1 assurance. The ORACLE Trusted DBMS has roles defined as collections of traditional DAC constraints. An objective on the LOCK DBMS program was to integrate the two approaches for defining roles to provide TDBMS roles enforced in a mandatory, A1 manner. As noted earlier, this was achieved to a certain degree by creating separate instances of the TDBMS for specialized roles. The roles provided were a read-only role and a read-write role.

In many ways providing high assurance role enforcement in a DBMS is just as difficult as providing high assurance DAC. Consider the current trend in DBMS technology towards multi-threaded servers. Since the server is executing for several users simultaneously, the server is responsible for maintaining user (and role) accountability. A possible approach for higher assurance would be to have just one server for each role, but this requires a mechanism for connecting a client to the proper server and implies the overhead of one instance per server as discussed previously.

If a single multi-threaded server is used and the various user actions being performed by the DBMS are individual threads, what is needed for a strong policy is separation between threads. Since Type Enforcement is currently done by creating separate subjects (processes) in different domains, it does not provide separation at the correct granularity. Modifications to LOCK's current Type Enforcement mechanism might be possible that would not require separate subjects be created before the mechanism can supply separation. Conceptually, this could be done by creating a small intermediate TCB subset that provides a finer granularity mechanism on top of the Type Enforcement mechanism. This remains an area for future research.

## Summary

The LOCK DBMS program showed that it is possible to use Type Enforcement to increase the security and integrity of a COTS TDBMS by providing strong separation of the DBMS system from the rest of the system and by allowing the TCB to enforce certain roles. However, the underlying architecture of Trusted Oracle limited the degree to which Type Enforcement could be used to enhance the system's assurance.

## References

[1] National Computer Security Center. *Trusted Computer Systems Evaluation Criteria (TCSEC) - DoD 5200.28-STD*. December 1985.

[2] Oracle Corporation. *Trusted ORACLE Administrator's Guide*. Oracle Corporation, Redwood City, CA. 1992

[3] Secure Computing Corporation. *System Specification for the LOCK DBMS Program*. Secure Computing Corporation, Roseville, MN. 1993

[4] National Computer Security Center. *Trusted Database Interpretation of the TCSEC (TDI)- NCSC TG-021*. August 1990.

[5] W.E. Boebert and R.Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies", *Proceedings of the 8th National Computer Security Conference*, NBS, 1985, pp. 18-27.

## Discussion: Architectures

**Discussion Leader: Rae K. Burns, AGCS, Inc.**

(paper not available)

## ASSURANCE

# A Position Statement: High Assurance DBMS

Rae K. Burns
AGCS, Inc.
91 Montvale Ave.
Stoneham, MA
(617) 279-2864

Current DBMS technology has evolved to the point where low assurance (C2/B1) MLS DBMS products are available and B2 designs have been developed. However, with the availability of additional high assurance operating systems (e.g., LOCK, TMACH), the need for a high assurance DBMS becomes more evident. For example, the MISSI program, which is using high assurance workstations as a foundation for multilevel operation, could benefit substantially from high assurance DBMS products designed to integrate securely with the trusted operating system. However, there is still no accepted approach for a high assurance MLS DBMS for a B3 environment.

## Previous Approaches to MLS DBMS

To date there have been two basic approaches: the trusted subject approach and the Schaefer-Hinke (SeaView) approach. The advantages and disadvantages of each have been well debated.

| Approach | Advantage | Disadvantages |
|---|---|---|
| Trusted Subject | DBMS enforces the MAC policy and can support a large number of sensitivity levels. It can make trade-offs to balance integrity requirements and secrecy requirements | The DBMS must have an OS privilege to violate the MAC policy. For high assurance, this extends the scope of covert channel analysis and penetration testing to include the DBMS |
| SeaView (Schaefer Hinke) | The DBMS is constrained by the OS MAC policy; the high assurance of the OS is not compromised | The database must be subdivided by sensitivity level and accessed by different instantiations of the DBMS. The DBMS cannot enforce multilevel integrity constraint. Also, there can be no use of a trusted path for DBMS operations since it is not part of the TCB. |

Some questions need to be addressed:

1. Are there other hybrid approaches that might mitigate some of the disadvantages of each approach?
2. Do the microkernel systems (e.g., TMACH, SYNERGY) offer a better base than traditional architectures for a high assurance DBMS?
3. Is DBMS technology moving toward DBMS architectures that would be a better match for a B3 system (e.g., a more compact "kernel" that could be trusted)?
4. Do object-oriented DBMS architectures provide additional alternatives that could be exploited for high assurance?

## Hybrid Approaches

It may be possible to combine the two traditional approaches to minimize the amount of trusted code in the DBMS. For example, if separate OS files were used, then any operations that SELECTed data could be untrusted; update operations could be still be performed by trusted code to assure that integrity constraints are enforced. To support a trusted path for SELECTs, there would still need to be a DBMS TCB component that could be used to access a database for SELECT, but it might not need to support a full set of query processing operations. Other hybrid solutions might be feasible depending upon the features of the OS TCB.

## Microkernel Architectures

The microkernel architectures separate policy and enforcement mechanisms more clearly than in traditional TCB architectures. The kernel enforces primitive policies and relies on different servers to provide the policy interpretation and enforcement of the policy on system resources. In this type of system, a DBMS server concept fits well, but still requires minimization to meet B3 system architecture requirements. Figure 1 illustrates a possible architecture.

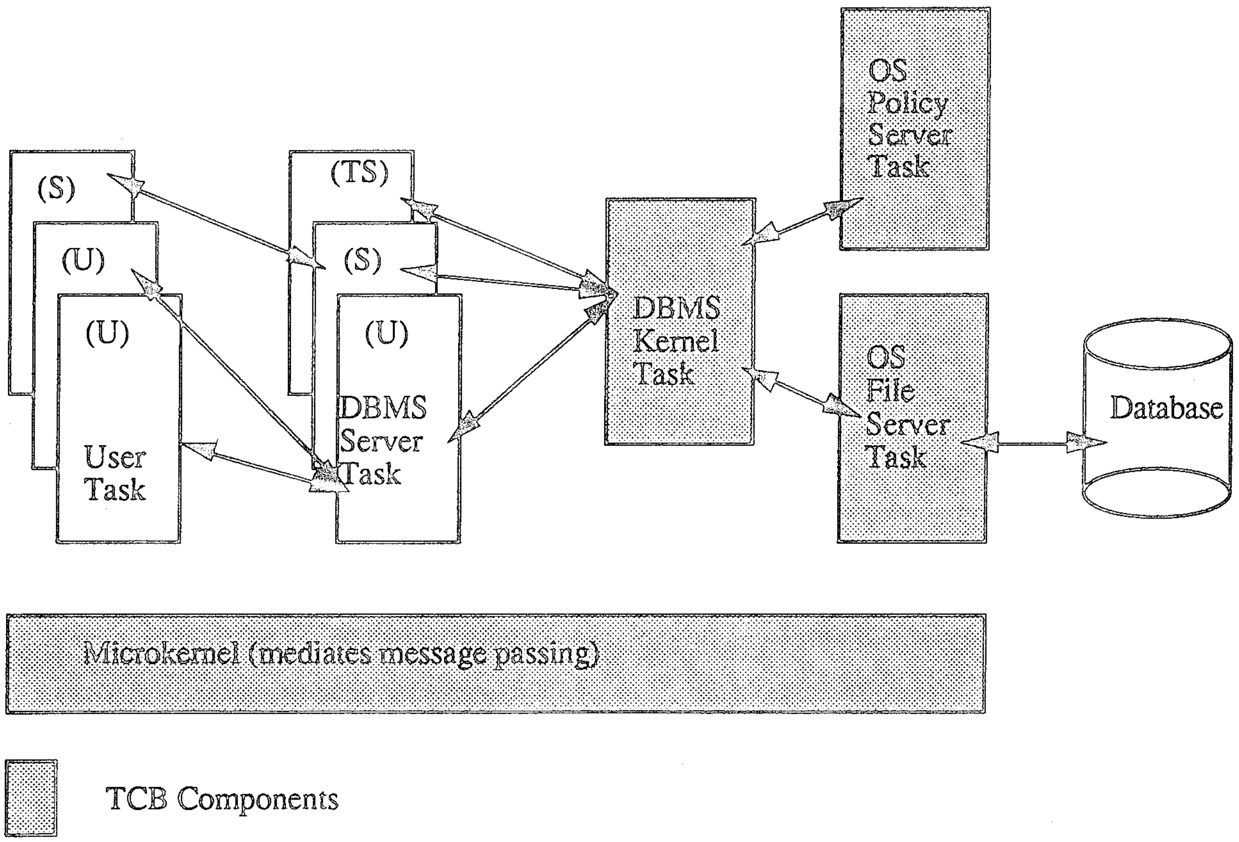


Figure 1. Microkernel Architecture

## DBMS Architectures

A major issue for a high assurance DBMS has been the complexity of the software that must be in the DBMS "kernel." Typically, a DBMS kernel supports not only I/O operations but also performs transaction management, concurrency controls, integrity constraint enforcement, and, in some cases, complex query processing. Since DBMS performance is a major concern, the security mechanisms generally have not been implemented using a "conceptually simple" mechanism; they have been implemented with techniques to avoid the addition of performance bottlenecks. However, research to identify minimal DBMS kernel functions could extend the microkernel concepts into the DBMS arena.

## Object-Oriented DBMS

While current OO DBMS products are primarily derived from OO programming concepts, the object-oriented paradigm offers potential for new DBMS architectures. By combining the message passing paradigm of the microkernel architecture with a DBMS based on an object model, it may be possible to support the minimization of TCB functions that is essential for high assurance.

## Summary

The concepts that underlay the microkernel operating system and that form the basis of the object-oriented model may also be applicable to high assurance DBMS architectures. They offer potential for resolving some of the issues that affect the more traditional approaches to multilevel database management. With the advent of this new technology, it is important to investigate how it might be used to move multilevel database technology to higher assurance solutions.

## Discussion: Assurance

**Discussion Leader: Ravi Sandhu, George Mason University**

(paper not available)

## DISTRIBUTED/FEDERATED SECURE DATABASE MANAGEMENT SYTEMS

# Contracts for Data Sharing among Autonomous Organizations

Catherine D. McCollum
The MITRE Corporation
7525 Colshire Drive
McLean, Virginia 22102

mccollum@smiley.mitre.org

## 1. Introduction

Organizations' computing resources are increasingly being interconnected into large-scale distributed systems. Distributed database capabilities for flexibly and conveniently retrieving data stored in different databases, on separate network nodes and perhaps in different DBMSs or file systems offer a great deal of power for gathering and integrating data from different sources in such an interconnected system. This ability to access dispersed databases is key to allowing organizations to share data to carry out a joint or common mission.

However, such large distributed systems are seldom fully under administrative control of any one organization. Cooperation among organizations under different decision-making authorities requires that distributed database functions be carried out in a way that respects the autonomy of the constituent organizations (and their database systems).

A specific area where cooperating organizations may need to retain decision-making authority is in controlling access to and protection of their sensitive data. Since different organizations operate under different requirements for protection and control of their data, security capabilities must be available that can support these differences. To be able to share data with others, each organization must be able to arrange for its own data to be appropriately protected even when used or stored on a system belonging to another organizational entity. This implies a different kind of controls than those which are following naturally from distributed systems extensions of database technology (e.g., ascertaining the identity of a user across the network to support checking of direct access authorizations). Additional controls are needed that can represent agreements among the separate, sovereign organizations concerning conditions under which they are willing to share data with the other organization and its users, and the protection and handling responsibilities that are incurred in return.

## 2. Federated Database Systems, Autonomy, and Security

Federated database technology, which is designed to provide distributed database capabilities under conditions of decentralized control, is particularly attractive as a basis for data protection in these circumstances. A fundamental goal of federated database systems is autonomy: cooperation without sacrificing independence. This is the characteristic that distinguishes them from other distributed database systems. Federated data management seeks to allow partial, controlled sharing with negotiated coordination of shared activities, while minimizing the role of any centralized authority [Heimbigner 1994]. The goal of autonomy must of course be tempered somewhat, because a certain degree of cooperation is necessary simply to carry out the mechanics of sharing data. And, when the systems handle data of any sensitivity, the desire for highly autonomous interaction must in addition be balanced against the responsibility to protect the security or privacy of the data. (In fact, though DBMSs participating in a federated DBMS are commonly referred to as being autonomous, it is perhaps more accurate to identify them as semi-autonomous [Oszu and Valduriez], since, although they can operate independently, some modification is needed to allow them to cooperate in executing distributed requests, and their acceptance of this limited loss of freedom is indicated by the fact of having joined the federation. Full autonomy more accurately applies to multidatabase systems, where the DBMSs are unaware of each other.)

Much work is going on in distributed transaction management and other areas necessary to the operation of a distributed or federated database system, but the question this work leaves unanswered is what is required for these organizations to come to the point where they are willing to share data in the first place. Concern about what happens to data when access is extended to outsiders not under the same jurisdictional control and the data perhaps may be imported into other systems is a significant barrier to the formation of federations and sharing of data. This is the area on which our work focuses.

We are aware of only limited work in the area of security for federated database systems. [Morgenstern et al. 1992] presents some initial ideas concerning architectures, a reference model, and issues for multilevel and discretionary security within a database federation. [Idris et al. 1994] has looked at mapping security classes when integrating individual database schemas to a global schema in a multilevel secure distributed DBMS. [Jonscher & Dittrich 1993] reviews work applicable to discretionary security in federated database systems, including decentralized authorization, view-based approaches such as a two-tiered (federation- and local-level) access control list scheme, and logic-based approaches. They identify many security-related issues specific to the heterogeneity and autonomy demands of federated systems that are not addressed by existing work. [Nierstrasz et al. 1993] describes the framework for a system, CHASSIS, in which they are investigating issues raised in [Jonscher & Dittrich 1993]. The system is tightly coupled, in that it uses a global schema. CHASSIS places cooperating database systems (or systems providing other types of services) within an object-oriented interoperability framework that encapsulates the systems as "cells." Interaction among the cells is handled via type matching, object mapping, and connection trading that occurs at the boundary, or "membrane" of each cell, to determine the security requirements and other parameters. The federation, which implements a global policy, is itself a cell with respect to the rest of the system. The global policy provides discretionary security, with decentralized authorization based on ownership and grant flags, negative authorizations overriding positive authorizations, and an extensive role capability. Much of the focus is on resolving issues arising from heterogeneity of the individual systems discretionary security features.

## 3. The NAIAD System

We are taking a different approach to address the need for a secure federated data management system that will overcome barriers to sharing across jurisdictions. We are developing the Negotiated Agreements for Interaction of Autonomous Databases (NAIAD) system, which focuses on providing a functionally rich set of application-definable controls on exchange of data and handling of imported data, overlaid on typical commercial DBMS product features. Our emphasis is on creating and carrying out policies specifically on the exchange of data. We assume for the time being that heterogeneity is addressed elsewhere, for example, by heterogeneous distributed DBMS gateways and translators.

The approach to security in NAIAD is based on the concept of contracts, negotiated agreements that specify the conditions under which access will be allowed, and the obligations incurred in exchange for protection and handling of the data accessed or imported. This concept closely mimics the idea of negotiating and signing a contract in the human domain. Generally, when two organizations decide to cooperate, some form of memorandum of agreement is drawn up that sets out the privileges and responsibilities of each. Each in effect concedes a small part of its autonomy, agreeing to adhere to what is requested of it by the other in return for some desired result, such as access to information belonging to the other. Our contracts model works on the same principle.

The idea of contracts has been proposed for various uses in automated systems over the past several years (see for example [ISO 1984, Greenberg & Rathman 1990, Meyer 1992]), often in connection with integrity, determination of specific details of a particular interaction, or information hiding. It has also been noted that the idea of contracts is particularly appropriate within the federated database model. For example, [Alonso & Barbara 1989] discusses dynamic negotiation for access to data among autonomous nodes in a federated database system, but is primarily concerned with negotiating based on cost considerations; i.e., determining whether a replica will be established at the importer site for future querying and agreeing on the frequency of update to maintain the replica.

[Ahlsen et al. 1993] further developed the idea of contracts in database federations. Their contracts, which represent bilateral agreements between nodes in the federation, consist of terms on duration (initiation and termination conditions), object access (direct access versus creating a replica on the importer site, as in [Alonso & Barbara 1989]), availability (acceptable degree of deviation from expected behavior), authentication (procedure to verify contract signatories), and accounting (cost for establishment and use of the contract). The focus in this work is on a process for formation of contracts. Because they assume that the same data may be available at different sites, they structure the dialogue to establish a contract as follows: (1) an announcement phase in which a would-be importer advertises to the federation its desires for the data it would like to obtain and proposed conditions, (2) a bidding phase in which eligible nodes respond by submitting bids (effectively, counterproposals with the conditions under which they are willing to offer the data), (3) a negotiation phase in which the importer selects one of the bidders and the two successively modify their positions until they reach agreement, and (4) a commitment phase in which the sites affirm their intention to carry out the contract and create required contract objects.

NAIAD emphasizes, rather than the process of arriving at a contract, more complex policies that may be represented as a contract. Contracts in NAIAD define requirements for protection and handling of data. These requirements may include the traditional positive and negative authorizations on access to data, based on users, roles, context- and content-dependent conditions, but may also include an active element, such as a procedure that must be carried out in connection with access to the data. The latter correspond to what [Jonscher 1993] refers to as normative policies, or duties, and what [Moffett et al. 1993] refers to as imperatival policies. In including active, procedural controls, we are responding not strictly to the federated paradigm, but to what we see as cross-jurisdictional sharing needs. Both the authorization and active types of requirements are applied to create obligations concerning the importer's handling of data imported under the contract. The idea is that with a foundation of appropriate types of controls to ensure that data is handled properly by an importer, the system owning the data will be willing to extend more generous access than otherwise, under the condition that specified restrictions and procedures be followed. In the rest of this paper, we briefly present the main features of NAIAD.

## 3.1 Architecture

NAIAD employs a loosely coupled architecture, in that no global schema is maintained[1]. However, it is not as far to this end of the spectrum as the system described in [Ahlsen & Johannesson 1993], in which the nodes need not have any knowledge of all nodes in the federation, but just those with which they are directly acquainted. In NAIAD, the federation

[1] [Jonscher & Dittrich 1993] refers to a choice between tightly coupled (global schema, location transparency) and loosely coupled (no global schema, no location transparency). Our view falls somewhere in between: each database system advertises its export schema, and any component system can use the collection of these export schemas (the parts accessible to it) to resolve unambiguous references. References that are ambiguous can be resolved either by policy (e.g., assume local version unless instructed otherwise) or by reference to the user.

provides a base level of system capabilities and inter-organizational trust upon which the members can rely. The reason for this basic level of trust is that the federation defines a domain within which all the members are known, all have met some set of membership requirements (in terms of system capabilities), and all have agreed to some set of groundrules related to the overall mission for which they are cooperating.

Within the federation, pairs of individual nodes can enter into contracts to share specific data. Negotiation of acceptable contracts occurs in the human domain, but once agreed upon, a contract is incorporated into the system as an automated specification. These contract specifications explicitly document the mutual expectations for control of data under the sharing arrangement. The contracts specify initiation conditions, the specific scope of data involved, roles that users at each node may play with respect to the interaction, specific obligations that the owner imposes on the importer as a consequence of accessing or importing the data, and termination conditions. Initiation and termination conditions specify events that trigger the contract's going into effect or being terminated (such as the arrival of a particular date and time), and any required actions that must take place at that time. Roles in the contract are specific to that contract and may be different from roles defined at the level of the individual database system. The obligations may include restrictions constraining access that will be allowed (either *in situ* or after transfer to an importer site) and may specify procedures that must accompany access to the data. The latter means that the policy expressed by a contract may have an active element, that defines required actions the importer must undertake in connection with the exchanged data.

### 3.1.1 Schema architecture

The schema architecture for NAIAD is simple. As shown in Figures 1a & 1b, we describe it in comparison to Sheth and Larson's five-level reference schema architecture, which includes local, component, export, federated, and external schemas [Sheth & Larson 1990]. Because we have submerged the issue of heterogeneity for now, we do not distinguish between a

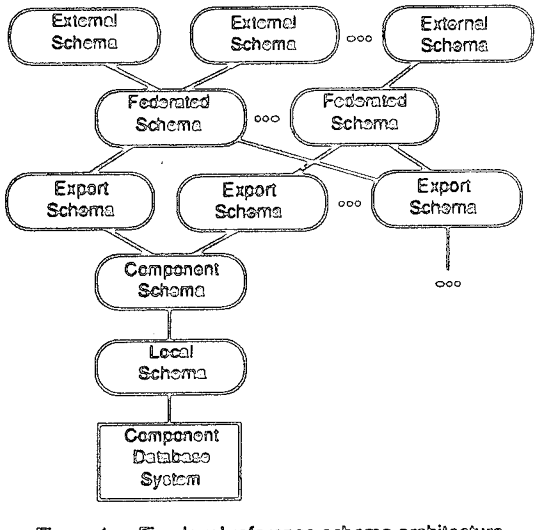


Figure 1a. Five-level reference schema architecture

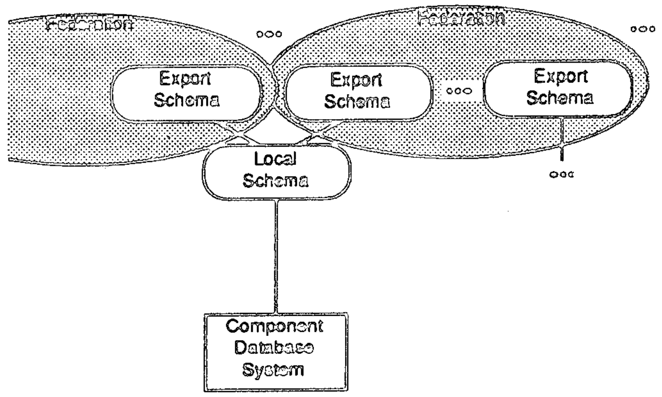


Figure 1b. NAIAD schema architecture

component (or constituent) schema and the local schema of the database system, as do Sheth and Larson. Above its local schema, each database system within the federation has an export schema with respect to that federation. A node may belong to more than one federation (provided that it meets the membership requirements of each), and will have a separate export schema for each federation to which it belongs. A node's export schema, however, does not offer access to the database objects it identifies uniformly to all members of the federation. Although some data objects may be offered "publicly" within the federation without further constraints, others are available only to nodes which have specifically contracted for access to them and agreed to all the strings attached. Thus, the node's export schema represents the data which the node is making available to the federation collectively, but, individually, other nodes may have access only to part of it.

There is no explicit federated schema. (Of course, the collection of all the individual export schemas could be considered a notional federated schema, but there is no federation-level integration of schemas). This in turn means that there are no external schemas on the federation schema. Data imported into a node from other members of the federation may be represented as additional objects in its local schema, and external schemas may exist over this local schema (essentially in parallel to the schema it exports to the federation), but these schemas, if they exist, are immaterial to NAIAD.

NAIAD does, however, require that some federation information be known globally. All members must know what sites are federation members, what the members' capabilities are, and what the federation groundrules are. This information could either be fully replicated throughout the federation, with some protocols for updating it when new members are admitted, or could be maintained on a separate federation server, which then could also serve as a membership gatekeeper.

### 3.1.2 System Architecture

NAIAD's system architecture is also quite simple. The architectural goals are that it be transparent to users and applications of the local system, and that it be usable with off-the-shelf DBMS products. The former means that queries are submitted in the same manner as usual on the local database systems, and the latter means that the federation layer of NAIAD should not interfere in the internal workings of the local DBMS.

As shown in figure 2, there are two top-level components, a Contract Manager and a Distributed Data Manager. The Distributed Data Managers are the local DBMSs belonging to the sites. Whether the local DBMSs themselves have distributed capabilities

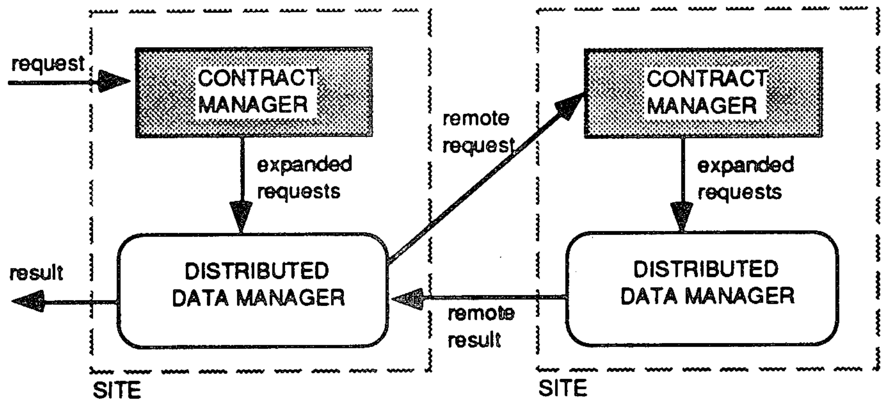


Figure 2. NAIAD System Architecture

is not critical; if they are absent, either the applications which submit database requests may have distributed knowledge, or the local DBMS can be supplemented with a distributed data management layer. A Contract Manager must be present on each database system participating in a federation. The Contract Manager maintains a database of the contracts to which the site has committed and implements the agreed-upon access restrictions and obligations of the contracts. Collectively, the Contract Managers can be viewed as providing a distributed reference monitor and situation monitor (in the terminology of [Moffett et al. 1993] ).

The key aspect of the architecture is that the Contract Manager intercepts all requests which would ordinarily be submitted directly to the local DBMS. Whether this is easily done in the case of a particular local DBMS product depends on its capabilities. If the DBMS provides retrieval and update triggering, it would be very easy. If these facilities are lacking, other methods would need to be used to place the Contract Manager in the stream of input to the DBMS.

When a database request is submitted, then, the Contract Manager examines it before it reaches the local DBMS. Based on the data and operation requested, it determines whether any contracts apply to the request. If no contracts apply, the request is forwarded for processing to the local DBMS, where it is checked against the DBMS's native authorizations and executed. If there are applicable contracts, the Contract Manager checks the request against contract-defined authorizations and then expands the request. Expansion may replace or modify the original request (for example, to be more restrictive) and may add database actions (such as updating a log relation) in addition to the original or modified request. The expanded request (which may be a "script" containing multiple requests) is then submitted to the DBMS.

When the DBMS evaluates the query, if it determines that some of the relevant data is located at another site, it may generate requests which are sent to the remote site. At the remote site, these requests also must be intercepted by the Contract Manager at that site and subjected to authorization checking and expansion.

## 3.2 The Contract Lifecycle

Contracts, as defined in NAIAD, are static. Once installed and activated, they apply to all subsequent access attempts within the scope of data and operations identified by the contract, until they are suspended or terminated. Changes to the contracts require re-negotiation and reinstallation. We identify the following states in the contract lifecycle:

1. proposed (under negotiation)
2. ratified (agreed to by both parties)
3. installed (incorporated into the automated system and awaiting the occurrence of the event specified as the initiation condition)
4. initiated (in operation for database accesses)
5. suspended (temporarily disabled)
6. terminated (permanently disabled).

States (1) and (2) are outside the scope of the current system, since we are not attempting to support dynamic negotiation within the system. Negotiation of an automated contract would probably correspond to negotiation of a similar agreement in the human domain, such as a memorandum of understanding. Because the policies represented by contracts are complex, dynamic negotiation would be quite complicated. We limit our attention to static policies negotiated outside of the automated system operation but which still can provide significant useful capabilities. We are, however, interested in looking in the future at what sort of automated tool support could be provided to assist with composition, negotiation, and validation of the automated contracts.

Installation of the contract can be done independently on each participating site. Initiation, suspension, and termination, on the other hand, may require some protocol for coordination among the sites, to avoid the occurrence of situations where one site believes that a contract is in effect and behaves accordingly while the other does not.

Further, when a contract is suspended or terminated, it will be necessary to distinguish between requests that were received before the interruption but still have associated processing in progress (for example, a contract might have delayed effects, such as requiring that an imported data object be deleted after 10 days) and those received after the suspension or termination. Actions associated with execution of the contract for a request that arrived before the suspension would be allowed to proceed, while new requests would not invoke the contract but would instead be passed through to the local DBMS and be checked instead against its native authorizations. Typically, these native authorizations would be more restrictive than those provided through the contract (since additional controls and procedures are not being followed), and the request might be rejected.

## 3.3 Contract Language

Contracts are expressed in an extended version of SQL, the prevalent relational database language. The main modification is to define an extended set of events on which triggered actions can take place. In addition to triggering on update events, we include triggering on retrievals, time-based events (absolute or interval), and receipt of messages.

In addition, several types of contract-specific information are represented in the contract specification, to declare the contract-specific roles and authorizations, to define the scope of data covered by the contract, and to specify recovery actions. We refer to these recovery instructions in the contract specification as "fine print". The recovery actions would be carried out if an execution of the contract fails for some reason, and might vary depending on the point in its execution at which the failure occurred. The normal execution of a contract may require that a series of actions be carried out over a period of time. The contract thus does not correspond conveniently to a traditional database transaction, in which resources are locked and partial results are not visible to other users until after the entire transaction commits and completes. The use of a nested or long duration transaction model appears more appropriate, and we are exploring the use of compensating transactions concepts from those models to handle recovery of a partially completed contract execution.

## 3.4 Conflicts

Sites incur obligations with each contract they sign. If two contracts apply to the same or overlapping data, it is possible that they might specify inconsistent or incompatible rules for access and handling of the data. The rules implemented in contracts are of three types: permissions ("cans"), prohibitions ("must nots"), and actions ("musts"). If one contract says that an operation or action must not occur, and another says that it must, we refer to the resulting impasse as a conflict.

We are currently investigating sufficient constraints that could be imposed on the definition of contracts to be able to guarantee that a set of contracts will be free of conflicts. However, it may be impractical to eliminate all conflicts for two reasons: first, the conflict-free constraints may be too restrictive in practice (requiring that data referred to by the contracts be disjoint, for example), and second, data owners may be willing to accept the risk of an infrequent conflict (for example, a content-sensitive one that depends upon values arising in a data object that meet two different conditions not usually encountered together) in exchange for the ability to define the contracts more flexibly. Because of this, we do not assume that system operation is free of conflicts. Conflicts may be resolved through a variety of simple strategies, such as use of a priority scheme, in which a priority is assigned to a contract at the time when it is defined. For example, contracts whose failure to execute would be catastrophic would be given a higher priority than those which failure would create a nuisance-level anomoly. When the impasse is encountered, the execution of the contract with lower priority would be aborted, and steps defined in the contract fine print would be taken to recover.

## 3.5 Execution model

A high level view of the NAIAD execution model is given in Figure 3. A submitted request undergoes contract-specific authorization checking before being expanded. A request that

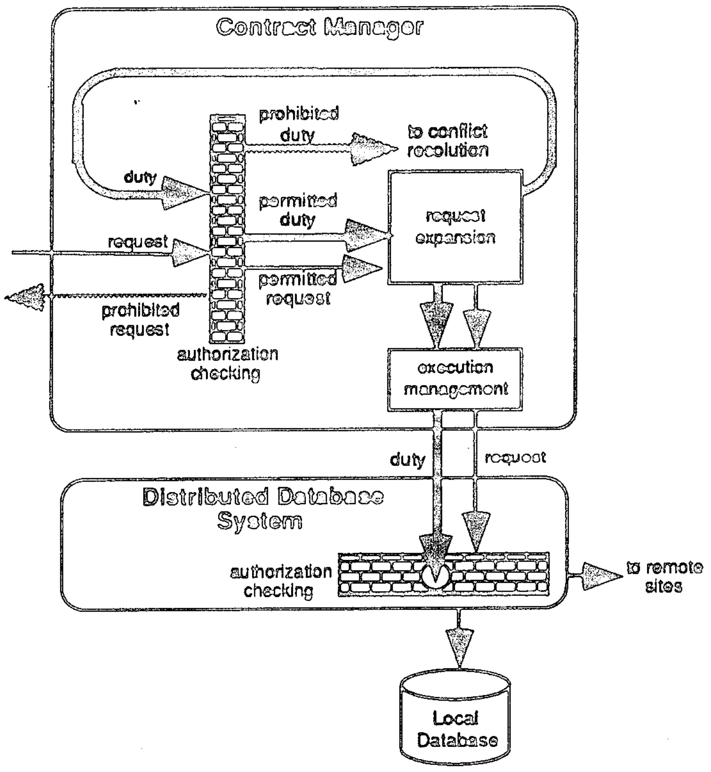


Figure 3. Execution model

does not fall under the terms of any existing contract remains an ordinary database request and is simply forwarded to the local DBMS for processing. A request that is found to be subject to a contract is expanded to produce a set of "duties," which are themselves database requests. The reason we distinguish them as duties rather than simply requests is that once it has been determined that a request is permitted and falls under the terms of a contract, it becomes obligatory to carry out all of the steps identified in the contract. Next, each of the new duties must itself undergo authorization checking and expansion, in case it also falls within the scope of a contract. A duty that runs afoul of the authorizations of another contract cannot simply be rejected, since it is part of a set of actions the site has committed itself to carry out, so it is sent for conflict resolution. A permitted duty is checked to see if it must itself be expanded. (To prevent endless looping through the expansion step, an annotation provided in the contract language can be used to signify the point at which a duty should not be expanded further.)

When the expansion process has bottomed out, each duty is forwarded to the execution management function, which coordinates with the local DBMS to execute it. Here also, there is a distinction between ordinary requests and those executing as duties. Since duties result from the existence of a contract, in which a site has agreed to carry out special conditions and procedures set by the data owner and is in return extended special access to the data, a duty executes with greater privilege (signified in the diagram by the hole in the DBMS authorization checking). In addition, the execution of duties by the local DBMS is carefully monitored by the execution management function of the Contract Manager, because if a duty fails for any reason, including ordinary system exceptions, recovery according to the contract which generated it must be carried out, and any other contracts in the expansion tree may also need special recovery.

## 4. Conclusion

In the NAIAD system, we have defined a framework for the definition and enforcement of very flexible controls, including both traditional authorizations and prohibitions, and active, procedural controls, in a loosely coupled architecture. These controls are implemented as contracts, which mimic a familiar mode of cooperation in human interaction. The contracts model, along with the architecture we have developed, supports the creation of very flexible, application-specific controls and allows the individual database systems to retain a high degree of authorization autonomy.

However, we do not wish to minimize the difficulty of this problem, and there are many significant issues that need to be worked out. For example, the whole notion of controlling data once it has been imported into another system presupposes that there is some method available of not only segregating or marking the data with its owner but also of constraining the flow of the data into other containers. In practice, in many environments this is done through trusted applications (not in the multilevel security sense) that allow only very restrictive functions in the user interface, but a complete solution might require the incorporation of a propagated authorization model such as [McCollum et al. 1990]. Many other issues need signicant research. One is the question of how to analyze sets of contracts to identify conflicts, particularly because actions within the contracts could modify authorizations in response to events, creating a dynamic authorization environment. Another is the question of how the two parties to a contract are each to satisfy themselves that the other's contract specification, as translated to executable mechanisms of the DBMS, is accurate and being reliably enforced. In addition, there are many semantic and implementation questions, such as how the system can detect and respond to side-effects and sub-requests (such as firing of database triggers or nested queries) without being more closely integrated with the local DBMS. Also, it is not clear to what extent the Contract Manager can be implemented generally and avoid being tied to a specific DBMS or class of very similar DBMSs. Finally, research will be needed to discover how NAIAD can incorporate support for heterogeneity in security models and mechanisms and be placed in the broader context of distributed object management.

Nevertheless, we believe that this model has the potential for filling an important need. More and more, organizations are under pressure to provide means of sharing data in support of broader missions. At the same time they must continue to meet requirements for protecting the security and privacy of data, which may differ from one organization to another and vary according to the type of data. The need for distributed data management technology brings with it a need for defining and enforcing appropriate protection of data being accessed across jurisdictional boundaries. Cooperation of independent jurisdictional entities with minimal loss of autonomy is not fully served by existing distributed database authorization models. Our initial experience with prototyping these concepts indicates that, though the current implementation is quite limited and crude, it is capable of providing interesting and useful controls on the exchange of data in a cross-jurisdictional environment.

## References

[Ahlsen 1993] M. Ahlsen and P. Johannesson, "Contracts in Database Federations," Stockholm University, ISBN 91-7153-101-7, March 1993.

[Alonso & Barbara 1989] R. Alonso and D. Barbara, "Negotiating Data Access in Federated Database Systems," in Proc. Fifth International Conference on Data Engineering, Los Angeles, California, IEEE Computer Society Press, 1989.

[Greenberg & Rathman 1990] I. B. Greenberg and P. K. Rathman, *Distributed Database Integrity*, Final Report, SRI International, Menlo Park, CA, 1990.

[Heimbigner &McLeod 1985] D. Heimbigner and D. McLeod, "A Federated Architecture for Information Management," ACM Trans. Office Inf. Syst. (July 1985), 3(3): 253-278.

[Heimbigner 1994] D. Heimbigner, "Infrastructure for Federated Software Environments," presentation to NIST, 18 March 1994.

[Idris et al. 1994] N. B. Idris, W. A. Gray, and M. A. Qutaishat, "Integration of Secrecy Features in a Federated Database Environment," in *Database Security, VII: Status and Prospects* (T.F. Keefe and C.E. Landwehr, eds.), pp. 89-108, North-Holland, 1994.

[ISO 1984] International Organization for Standardization, *International Standard ISO 7498, Information Processing Systems -- Open Systems Interconnection -- Basic Reference Model*, ISO 7498-1984 (E), American National Standards Association, New York, 1984.

[Jonscher 1993] D. Jonscher, "Extending Access Control with Duties Realized by Active Mechanisms," in *Database Security VI: Status and Prospects* (B. M. Thuraisingham and C. E. Landwehr, eds.), pp. 91-111, North-Holland, 1993.

[Jonscher & Dittrich 1993] Dirk Jonscher and Klaus R. Dittrich, "Access Control for Database Federations, A Discussion of the State-of-the-Art," DBTA Workshop on Interoperability of Database Systems and Database Applications, Fribourg, Switzerland, October 1993.

[McCollum et al. 1990] C. McCollum, J. Messing, and L. Notargiacomo, "Beyond the Pale of MAC and DAC -- Defining New Forms of Access Control," in Proc. IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1990.

[Meyer 1992] B. Meyer, "Applying 'Design by Contract,'" IEEE Computer, October 1992.

[Moffett et al. 1993] J. D. Moffett, D. Jonscher, and J. A. McDermid, "The Policy Obstacle Course: A Framework for Policies Embedded within Distributed Computer Systems," SCHEMA/York/93/1, Department of Computer Science, University of York, UK, 1993.

[Morgenstern et al. 1992] M. Morgenstern, T. F. Lunt, B. Thuraisingham, and D. L. Spooner, "Security Issues in Federated Database Systems (Panel)," in *Database Security, V: Status and Prospects* (C.E. Landwehr and S. Jajodia, eds.), pp. 131-148, North-Holland, 1992.

[Nierstrasz et al. 1993] O. Nierstrasz, D. Konstantas, K. Dittrich, D. Jonscher, "CHASSIS -- A Platform for Constructing Open Information Systems," Proc. AFCET '93, Versailles, France, pp. 153-161, 1993 (French version);

also published in: Visual Objects, D. Tsichritzis, ed. Université de Genève, Centre Universitaire d'Informatique, pp. 235-245, 1993 (English version).

[Sheth & Larson 1990] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," ACM Computing Surveys, Vol. 22, No. 3, 1990.

# MUSET AND MULTILEVEL DATABASE TRANSACTIONS [1]

LouAnna Notargiacomo and Ken Smith

The Mitre Corporation
McLean, Virginia

## 1 INTRODUCTION

Users in a multilevel environment typically are cleared to access data labeled up to a particular sensitivity level, and they have an operational need to read and write data at all security levels their clearance dominates. Although MLS relational DBMS products are beginning to appear on the market (Informix, 1993; Oracle, 1992; Sybase, 1993), a limitation of these products is that to ensure mandatory access control enforcement (MAC), an application should only use untrusted processes executing single-level transactions. Each single-level transaction is assigned a security level; a transaction can read data at its level or below, but write data only at its level. This does not support applications that require both the ability to write at various levels and the database integrity enforcement provided by transactions. These products do allow for the execution of multilevel transactions, but only if privileges are turned on to allow for the bypass of some or all mandatory access control (MAC) enforcement.

These DBMSs also provide some homogeneous distributed data management functions (including the necessary primitives for a distributed commit protocol) and, in some situations, these products can be configured to allow the execution of multilevel distributed requests, again only if privileges are turned on to allow for the bypass of MAC enforcement.

In the MUSET effort, we are concentrating on the development of a generic multilevel distributed transaction execution capability that can be used in a variety of multilevel system configurations (Blaustein, 1993a). This research assumes a multilevel secure (MLS) distributed environment in which the nodes may operate at various accreditation ranges. This includes single-level nodes, multilevel nodes with some overlap in accreditation ranges, and nodes with disjoint accreditation ranges, all connected by an MLS network capability. An example configuration can be seen in Figure 1.

In executing a user's multilevel transaction, the goal of the MUSET design is to be able to transform a single multilevel transaction into a set of single-level subtransactions that can be executed in such a way that the result would be equivalent to a single-site, single MLS DBMS execution. In this way we can ensure that no remote subtransaction has to be executed with MAC privileges at a remote site. While it can be envisioned that a user would be able to execute a transaction with privilege at his local site, it is not appropriate to assume that the user would be granted these same privileges when requesting access to data at a remote site.

1 This work was funded by Rome Laboratory, under contract F19628-94-C-0001.

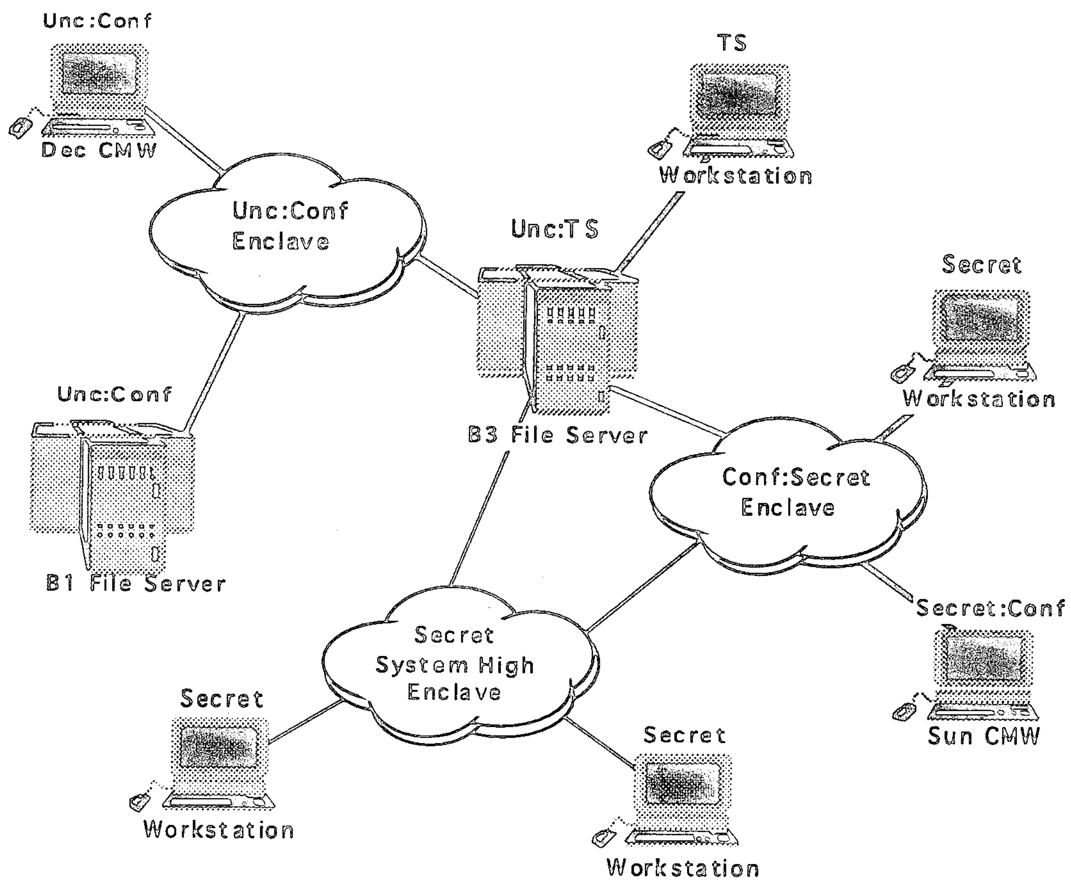


Figure 1. A MUSET Configuration

In order to allow for the distributed execution of multilevel transactions, several key technical problems need resolution. These include the ability to execute multilevel transactions atomically without the introduction of channels that can be used to violate security; the ability to execute these transactions so that they meet a high degree isolation[2], and the overall system can be returned to a stable state after a system failure; and the ability to protect the overall integrity of the database when data are distributed over multiple systems. In addition, it is critical that the developed distributed transaction algorithms work with the current suite of commercial relational untrusted and trusted products.

### 1.1 Related Work

Most of the work to date on transaction management for MLS DBMSs has focused on concurrency control protocols for executing single-level transactions concurrently. Work in (Jajodia, 1990; Costich, 1992a) has dealt with the execution of multiple single-level transactions, where the transactions can operate at different security levels. By definition, a single-level transaction has a fixed security level associated with it; the transaction can read

[2] At least degree 2 (Gray, 1993).

data that are at its level or below, but it can write data only at its level. (Keefe, 1990) has used a slightly more general model; his transaction can read data at or below its level, but can write to data that are at or above its level.

Some work has been done on concurrency control protocols for limited classes of multilevel transactions (which can perform reads and writes at multiple security levels). Transactions in Costich and McDermott (Costich, 1992b) can read and write data at multiple levels, but have the restriction that a transaction never writes to a lower level data item after accessing a higher level data item. This work assumed that the applications would be written in this low-to-high fashion before submission to the DBMS. The model adopted by Costich and Jajodia (Costich, 1992c) is most general and allow transactions to read and write freely at multiple security levels. This is done by introducing the notion of maintaining multiple versions within a single transaction. This work also was optimistic in that it assumes that any subpart of a transaction could always be executed successfully.

Although the ability to execute multilevel transactions concurrently is certainly desirable, it is more difficult to ensure that they can be executed atomically without creating illegal information flows. A major obstacle is that, if the portion of the transaction that is executing at the high security level fails, aborting the portion that is executing at the low security level would signal information.

In (Blaustein, 1993b), we developed a model of multilevel atomicity that defines varying degrees of atomicity and recognizes that lower security level operations within a transaction must be able to commit or abort independently of higher security level operations. This work utilized dependency graphs to identify the semantic dependencies between single-level sections of a multilevel transaction. It also used execution graphs as a tool for analyzing atomicity requirements in conjunction with internal semantic interdependencies among the operations of a transaction. Rules for determining the greatest degree of atomicity that can be attained for a given multilevel transaction were also provided. This work also developed several alternative transaction management algorithms that can be used to preserve multilevel atomicity when combined for the execution of multilevel transactions.

## 2 OVERVIEW OF MUSET

### 2.1 Mandatory Security Policy

MUSET's mandatory security policy is a distributed data management interpretation of the Bell and LaPadula model that supports both single-level and multilevel database applications. For all MUSET subjects, when a subject is created, it is assigned an executing range. A subject can be created by a user within a session or another subject and is assigned a range by its creator. The range of a newly created subject must be within the range of its creator.

A *session* is established with the trusted DBMS on database open. Each session is associated with the user who started it. A session has an access class range assigned by the user when it is created. The logon level of the user must dominate the upper bound of the session's range, and the lower bound of the session's range must dominate database low. Multiple transactions can be executed during a session. A *transaction* is a mechanism to structure application code so that sets of actions that impact a database occur atomically. A transaction has an access class range assigned by the user when it is defined. A *query* is a labeled data manipulation command that is contained within a transaction. The level of each query must be within the bounds of the transaction's range. The label of *language constants* (reserved words) is Unclassified. The label of data *object identifiers* (e.g., the name of an attribute within a relation) is equal to the label of that object's metadata. The label of a *variable string* can be specified by the user as part of the command syntax. If not specified, the sensitivity of a variable string defaults to the high of the transaction range.

The *read policy* is the same for all objects within a database. An object can be read only if the high of the subject's range dominates the level of the object. The level of the object's read must also dominate the level of the read command to prevent the flow of information from the command itself. A read request can specify a read range that must be within the allowed read range.

The *write policy* is the same for all objects within a database. An object can be written only if the level of the object is within the subject's range. The level of objects written must dominate the level of the writing command in order to prevent an illegal flow of information from the command to the object. A write command may specify a write range within the allowed write range.

## 2.2 Execution Scenario

Figure 2 presents an execution flow architecture that shows the major software processing modules that would be involved in the execution of a multilevel distributed database transaction. The MUSET system is being designed to be able to process multilevel transactions that are initiated from applications that are operating over a range of sensitivity levels. In all cases, where a range of levels is indicated, it is possible that the endpoints of the range are equivalent, in other words, the process is operating at a single-level. In the case of processes operating in system-high mode, we consider them to be single-level from a security policy enforcement perspective.

The user's application program first initiates a session with MUSET. This establishes the session sensitivity range. The session range must be within MUSET's processing range. Each multilevel transaction sent to MUSET must have a range that is within the bounds of the session range. Once the application session is established the application is free to transmit database transactions to MUSET. Each transaction includes the transaction range. In addition, each object within a transaction is labeled.

MUSET is trusted to operate over a range. The range corresponds to the operating range of the operating system on which MUSET executes. The single-level (SL) marking in Figure 2 indicates that if the software is running on a single-level (system-high) system, then the upper bound equals the lower bound, i.e., its running single-level.

When MUSET receives a multilevel transaction for execution, it is broken down into an equivalent set of single-level subtransactions or *sections*. This is performed by the MUSET process at the initiating application's site. The single-level sections are then sent to local or remote MLS DBMSs for processing.

This approach has been taken because current MLS DBMSs cannot execute multilevel transactions sent from a user's application unless the application is executing with the privileged to write down. Executing under this privilege disables the enforcement of the Bell and LaPadula *-property (prohibiting write-down operations). If the application is executing with the privilege to write-down and the application includes a multilevel distributed transaction, the MLS DBMS executes the multilevel distributed transaction in the same manner as it would execute a single-level distributed transaction. No special actions are taken to avoid invalid information flows from occurring. This opens the potential for significant illegal information flow. This can be avoided by use of the MUSET system. This is especially beneficial when a multilevel distributed transaction includes data accesses to remote sites, since the MUSET approach avoids the requirement to execute the remote sections as privileged processes.

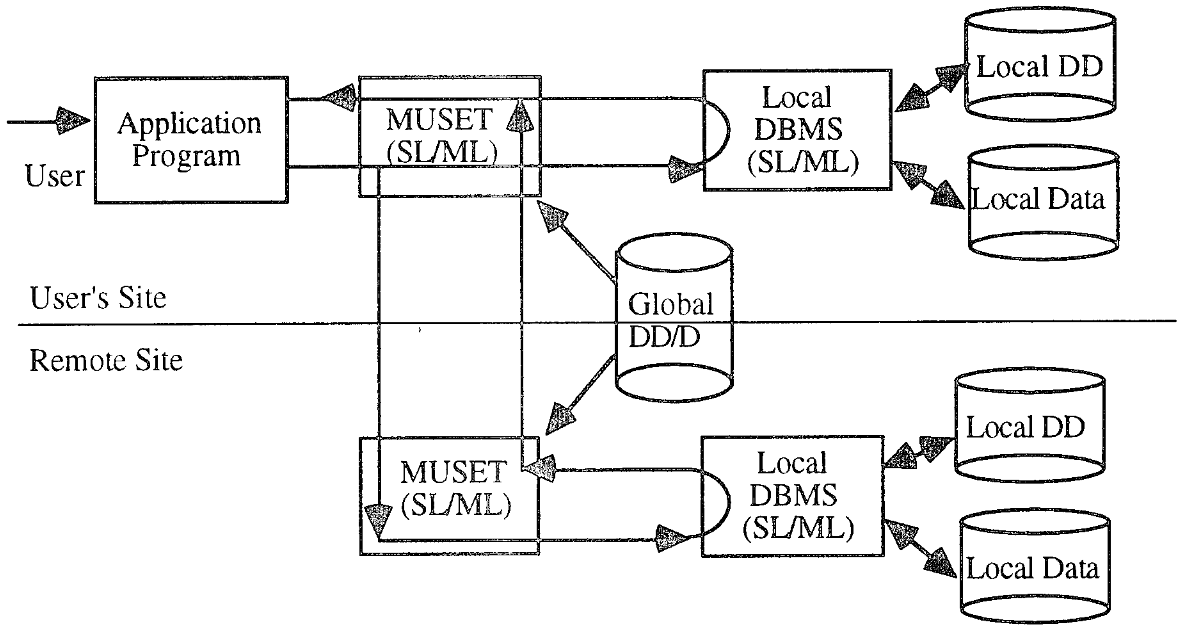


Figure 2. High Level Execution Flow Diagram

The DBMS processes that execute sections are either single-level or multilevel, dependent on the operating range of the system they run on. If the DBMS processes are multilevel, we assume that they only support single-level user application processes, for remote users. What we would like is for the combination of MUSET and MLS/SL DBMSs to be used (without privileges turned on) to execute multilevel transactions.

The MUSET architecture uses the concept of a global data description stored in a global data dictionary and directory (global DD/D). A separate global data dictionary exists for each logical distributed database. In addition, each is supplemented with global directory information containing the information needed to resolve the physical location of data objects. The MUSET layer derives the necessary information for transaction execution planning from the global DD/Ds. Each global DD/D contains the schema and directory information that describes all information within the range of its multilevel distributed database. A MUSET process will only be allowed to read the subset of global DD/Ds that it is authorized to access. Since a MUSET process operates over a range, it is authorized to read information dominated by its upper bound.

The global directory contains information about the physical distribution of data. For example, if a multilevel relation is fragmented across sites with different ranges, it will specify the range of the relation fragment stored at a particular host. The directory information also contains information needed to establish communication connections, for example host identifiers and host ranges (at least the range in common with this host).

The range of systems a MUSET process can communicate with is bounded by the range of the MUSET process. Information that is labeled below the range of the MUSET process is only accessible if the data resides on a system that's range intersects with the MUSET process' executing range. This also bounds the range of data objects that can be accessed through MUSET.

Single level sections are executed at the local or remote DBMSs and the data are sent back to the MUSET layer. All the responses are merged and formatted for output to the user.

MUSET's execution controller orchestrates the actual execution of sections through a *multilevel transaction execution protocol*. In a protocol, messages (such as "Precomitted," "OK to commit", and "release locks") are sent to and received from the component DBMSs, to attempt to ensure the combined execution meets correctness criteria defined in Section 3. Because of inherent conflicts between correctness criteria (such as security and atomicity), an important problem is designing execution protocols which provide a variety of approaches to making necessary correctness tradeoffs, satisfying the differing priorities of various users.

## 3 A FORMAL MODEL FOR MUSET TRANSACTIONS

In the following we define the basic building blocks of a formal model of MUSET transactions: multilevel transactions, multilevel schedules and execution protocols for multilevel transactions, and we define four desirable correctness properties for protocols and the schedules they generate.

### 3.1 Multilevel Transactions, Schedules, and Protocols

A multilevel transaction $T_i$ consists of a set of read and write operations, partially ordered by $<_i$, executed at multiple levels, where each obeys the Bell-LaPadula *- and simple-security properties. In a transaction $T_i$, operation $o_{ij}$ occurs at classification level j; thus $r_{ij}$ and $w_{ij}$ refer respectively to a read and a write at level j. When the transaction is clear from the context, $o_j$ is used. If there is an operation in $T_i$ at level j, we say j is *represented* in $T_i$. In general, two operations *conflict* if they both operate on the same data item and at least one of them is a write.

**Definition (Multilevel Transaction).** A *multilevel transaction* $T_i$ is a set of operations $o_{ij} \in \{r_{ij}[x_k] \mid x$ is a data item at security level k and j dom k$\} \cup \{w_{ij}[x_j] \mid x$ is a data item at security level j$\}$ that is partially ordered by $<_i$. All conflicting operations in $T_i$ must be ordered by $<_i$.

When a multilevel transaction is executed, events in the transaction are either aborted or committed and a record of its execution called a *schedule*, is generated. Furthermore, actions can be precommited to coordinate the execution of events in different sections (for atomicity purposes). In a transaction $T_i$, the *precommit* operation $p_{ik}$ means that all operations in $T_i$ at level k have been executed and there are no foreseeable barriers to committing. Therefore, a multilevel transaction *schedule* also includes events a (abort), c (commit), and p (precommit). Because single-level sections are assumed to be atomic, the results of write operations are not visible to other transactions until they are committed. Therefore, even *within* a multilevel transaction, writes at a lower level are not visible to higher levels until the lower level commit occurs.

**Definition (Multilevel Transaction Schedule).**[3] A *multilevel transaction schedule* $S_{T_i}$ (abbreviated $S_i$ when clear in context) for a multilevel transaction $T_i$ is a total order of operations, with ordering relation $<_{S_{T_i}}$ (abbreviated $<_{S_i}$) such that:

1. $S_i \subseteq T_i \cup \{p_{ij}, a_{ij}, c_{ij}\}$, where $p_{ij}$ is a precommit operation, $a_{ij}$ is an abort operation, and $c_{ij}$ is a commit operation.
2. $a_{ij} \in S_i$ iff $c_{ij} \notin S_i$.
3. If t is $c_{ij}$ or $a_{ij}$ (whichever is in $S_i$), for any other operation $o_{ij} \in S_{T_i}$, $o_{ij} <_{S_i} t$.
4. If t is $c_{ij}$ or $a_{ij}$ (whichever is in $S_i$), and $p_{ij} \in S_i$. then $p_{ij} <_{S_i} t$ and there does not exist an operation $o_{ij}$ such that $p_{ij} <_{S_i} o_{ij} <_{S_i} t$.
5. For conflicting operations $o_{ij}$ and $o_{ij}'$, $<_{S_i}$ must preserve the $<_i$ ordering.
6. If $w_{ik}[x_k] <_i r_{ij}[x_k]$, then $c_{ik} <_{S_i} r_{ij}[x_k]$.

Condition (1) defines the events in a multilevel transaction schedule. Condition (2) says this set contains a commit or an abort, but not both. Condition (3) says the commit or the abort (whichever is in this transaction) comes after all other events, and condition (4) states that a precommit, if present, comes immediately before the commit or abort with no intervening actions. Condition (5) states that the execution must preserve the order of conflicting operations within a level. Condition (6) states that when a read at a higher level follows a lower-level write of the same data item, the read operation must follow the commit of the write.

[3] The definitions in this section are based on the standard definitions for single-level transactions given in (Bernstein, 1987).

The criteria for a single-transaction schedule are used as the building blocks for schedules over sets of multilevel transactions.

**Definition (Multilevel Schedule).** A *multilevel schedule* $S_T$ over a set $T$ of multilevel transactions $T_1, T_2, ..., T_n$ is a total ordering of all operations in schedules $S_{T_1}, ..., S_{T_n}$, with ordering relation $<_{S_T}$, such that $<_{S_T}$ preserves the ordering of all conflicting operations within each $S_{T_i}$.

Note that when the set $T$ consists of a single transaction $T_i$, the multilevel schedule $S_T$ is identical to the multilevel transaction schedule $S_i$. In the following, we will use the term *schedule* for executions of both sets of transactions and single transactions.

There may be a large set of possible schedules for the same transaction or set of transactions. The notions of correctness discussed in the next section rely on the idea of *equivalent schedules*, defined here.

**Definition (Equivalent Schedules).** Schedules $S$ and $S'$ are *equivalent* iff for every database state D, executing $S$ in state D produces the same state D' as executing $S'$ in state D.

**Corollary.** Two schedules $S$ and $S'$ consisting of the same operations are equivalent iff every write that commits in $S$ also commits in $S'$ all pairs of conflicting operations are ordered the same in $<_S$ as in $<_{S'}$.

Transactions are executed (generating schedules) by means of a transaction execution protocol:

**Definition (Multilevel Transaction Execution Protocol).** A multilevel transaction execution protocol (or simply "protocol") $P$ transforms a set of transactions $T$ into a schedule $S_T$. We denote the transformation as $P(T) = S_T$.

## 3.2 Correctness Properties

Similar to the ACID (Gray, 1993) properties of standard database schedules, we define four "ACIS" correctness properties of multilevel schedules and protocols.

### 3.2.1 Atomicity

In a fully atomic (A-correct) schedule, all operations within a transaction must either commit or none may commit. In multilevel schedules, however, there is a commit or abort operation corresponding to each level at which writes are done. The following definition adapts the definition of atomicity to multilevel schedules.

**Definition 1 (A-Correctness).** Let $S_T$ be a schedule for a set $T$ of multilevel transactions. $S_T$ is A-correct iff for all schedules $S_i$ in $S_T$, if $\exists j$ such that $c_{ij} \in S_i$, then $c_{ik} \in S_i$ for all levels k represented by write operations $w_{ik} \in T_i$.

### 3.2.2 Consistency

Unlike traditional schedules, operations may need to be reordered in multilevel schedules to achieve the desired level of security. Consistency in a multilevel schedule requires the execution to preserve the order of all conflicting operations in the as-submitted multilevel transactions. We begin with a general definition of C-correctness and then make it more specific for multilevel schedules.

**Definition 2 (C-Correctness).** A schedule $S_i$ for a multilevel transaction $T_i$ is C-Correct iff it is equivalent to a schedule $S_i'$ such that for all conflicting operations $o, o' \in T_i$, $<_{S_i'}$ preserves the order of $<_i$.

C-Correctness requires the protocol to preserve the effects of the original ordering of all intra-transaction conflicts. While condition 5 of the multilevel transaction schedule definition ensures C-correctness within a security level, C-correctness must apply across security levels, too.

Since the basic security properties restrict the level of write operations to equal the level of the data item written, there can be no inter-level conflicts among write operations. Therefore, within a transaction $T_i$ there are two possible types of inter-level conflicts.

1. $r_j[x_k] <_i w_k[x_k]$, where j sdom k (called *read-first conflicts*)
2. $w_k[x_k] <_i r_j[x_k]$, where j sdom k (called *write-first conflicts*)

In a schedule $S_i$ for $T_i$, these conflicts translate into conflicts between $r_j[x_k]$ and the commit $c_k$. Therefore, the following corollary states that a schedule is C-correct if it is equivalent to a schedule in which the order of higher-level reads and lower-level writes within the transaction is preserved with respect to the higher-level reads and the lower-level commits within the schedule.

**Corollary.** A schedule $S_i$ for a multilevel transaction $T_i$ is C-correct iff it is equivalent to a schedule $S_i'$ such that for all conflicting operations $r_j[x_k], w_k[x_k] \in T_i$, where j sdom k, $r_j[x_k] <_i w_k[x_k]$ iff $r_j[x_k] <_{S_i'} c_k$.

### 3.2.3 Isolation

Our definition of isolation (I-correctness) is based on standard ideas of serializability (Bernstein, 1987).

**Definition 3 (Serial schedule).** Let $S_T$ be a schedule for a set $T$ of multilevel transactions. $S_T$ is serial iff for all transaction schedules $S_i$ and $S_j$ in $S_T$, if $\exists o_i \in S_i$, $o_j \in S_j$ such that $o_i <_{S_T} o_j$, then $\forall o_i' \in S_i$, $o_j' \in S_j$, $o_i' <_{S_T} o_j'$.

**Definition 4 (I-Correctness).** A schedule $S$ is I-Correct iff there exists an equivalent serial schedule $S'$.

### 3.2.4 Security

Multilevel transactions are required to conform to the simple-security and *-properties (Bell, 1975) which restrict information flows from higher to lower levels. However, the order of execution of operations within a multilevel transaction may themselves introduce security violations. For example, if operations at a higher level can cause an abort of operations at a lower level, this interference violates security -- an observer at the lower level could infer the existence of higher-level operations within the same transaction.

To capture this notion of non-interference, the S-correctness criterion, defined below, compares a schedule with operations at higher and lower levels to the same schedule with higher-level operations removed. If the results of the lower-level operations are the same in both schedules, then the higher-level operations did not interfere with those at the lower level. In the criteria that follow, we use the *purge* function to refer to schedules and transactions with higher-level events removed.

For a schedule $S_T$, a protocol P, and a set of multilevel transactions $T$, let the function purge($S_T$, L) return a new schedule which is $S_T$ with all events not dominated by level L removed. Similarly, let the function purge($T$, L) return a new set of multilevel transactions which is the set $T$ with the events not dominated by level $L$ removed from each member transaction.

Definition 5 (S-Correctness). Let schedule $S_T = P(T)$ for some protocol $P$ and set of transactions $T$. $S_T$ is S-correct with respect to $P$ iff, for all levels L, purge($S_T$, L) = $P$(purge($T$, L)) and no operation in $S_T$ must wait to start until an operation at a higher level of$S_T$ completes.[4]

### 3.2.5 Protocols

These correctness properties can also be applied to protocols:

Definition 6 (X-Correctness for Protocols). A protocol P is *X-Correct* for some correctness property X, where $X \in \{A,C,I,S\}$, iff for any set of multilevel transactions $T$, $S_T = P(T)$ is X-correct.

That is, P only generates X-correct schedules. In general, a subset of the letters A, C, I, and S are used to describe the subset of the properties that hold for a given schedule or protocol. For example, a protocol for which all properties but security hold is ACI-correct.

## 4 ONGOING AND FUTURE WORK

Correctness properties are not simply additive in protocols. Some combinations are provably impossible for any protocol. The strong form of S-correctness used interferes with the ability to attain other correctness properties. In particular, no protocol can be AS-correct, nor can

[4] Timing of operations also unltimately depends on issues such as system and network load. We ignore these factors here and only focus on timing dependencies between operations in schedules.

any protocol be IS-correct via a locking protocol (Smith, 1995). In this case, well-defined *partial correctness properties* can often be defined and achieved. A protocol partially correct for property X is designated $X^-$-correct.

Given the above limitations, the best possible protocols would either $ACIS^-$-correct or else $A^-CIS$-correct (obtaining I-correctness without using locking). In ongoing work, we are developing protocols for use in MUSET which meet the various provable upper bounds of correctness. We are also seeking protocols which are efficient and compatible with widely used COTS products, that also reach high levels of correctness. Future plans for the MUSET project call for the implementation of these multilevel transaction execution protocols in a MUSET testbed.

## 5 REFERENCES

Bell, D. E., and L. J. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation," Technical Report MTR-2997, The MITRE Corporation, Bedford, MA, 1975.

Bernstein, P. A., Vassos Hadzilacos, and Nathan Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.

Blaustein, B. T., Jajodia, S., Jones, V. E., McCollum, C. J., Notargiacomo, L., Smith K. P., and Rosenthal, A. S., *MUSET Multilevel Secure Distributed Database Management System*, MTR 93W0000236, The MITRE Corporation, McLean, VA, December 1993a.

Blaustein, B. T., Jajodia, S., McCollum, C. J., Notargiacomo, "A Model of Atomicity for Multilevel Transactions," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 120-134, May 1993b.

Costich, O., "Transaction Processing Using an Untrusted Scheduler in a Multilevel Database with Replicated Architecture," *Database Security, V: Status and Prospects* (Carl E. Landwehr and Sushil Jajodia, eds.), North-Holland, pp. 173-190, 1992a

Costich, O., and J. McDermott, "A Multilevel Transaction Problem for Multilevel Secure Database Systems and Its Solution for the Replicated Architecture," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 192-203, May 1992b.

Costich, O., and S. Jajodia, "Maintaining Multilevel Transaction Atomicity in MLS Database Systems with Kernelized Architecture," *Proceedings of the IFIP WG 11.3 Sixth Working Conference on Database Security*, Vancouver, BC, pp. 227-252, August 1992c.

Gray, J.,*Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, 1993.

INFORMIX Guide to SQL, April 1993

Jajodia, S., and B. Kogan, May 1990, "Transaction Processing in Multilevel-Secure Databases Using Replicated Architecture," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 360-368.

Keefe, T. F., and W. T. Tsai, May 1990, "Multilevel Concurrency Control for Multilevel Secure Database Systems," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 369-383.

ORACLE 7 Server: SQL Language Reference Manual, December 1992

Smith, K.P., B.T. Blaustein, S. Jajodia, and L. Notargiacomo, "ACIS Correctness Criteria for Multilevel Transactions," submitted to *IEEE Transactions on Knowledge and Data Engineering*, 1995.

Sybase Secure SQL Server Reference Manual, 1993.

## Discussion: Distributed/Federated Secure DBMS

**Discussion Leader: Catherine D. McCollum, The MITRE Corporation**

(paper not available)

## TECHNOLOGY TRANSITION

# SINTRA Technology Transfer: Lessons Learned So Far

*Judith N. Froscher*
*Oliver Costich**
Naval Research Laboratory
Washington, D.C. 20375

## 1 Introduction

Over the past year, NRL has been quite active in technology transfer efforts for SINTRA (Secure INformation Through Replicated Architecture) [1]-[13]. We will share some of our experiences during that time and point to considerations that can smooth the endeavor. First, we discuss how DoD information technology (IT) concerns and future plans should influence general security research and technology. Then we describe how the SINTRA concept supports these IT goals. The insight that we gained about the importance of replication in a cooperative, distributed computing environment allowed us to appreciate the fundamental role that the SINTRA approach can have in providing security for distributed computation.

Technology transfer is a chicken and egg problem. Vendors want to see user demand before they invest in commercializing new technology. Users want to see and use MLS systems before they move to them. This tension illustrates the differences that separate MLS technology from the mainstream of information technology. Users want to benefit from recent advances in information technology and are quite reluctant to accept MLS solutions because it is difficult for improved capabilities to be integrated into these systems. The SINTRA approach allows MLS technology and mainstream information technology to become alligned more closely.

Our goal is to commercialize MLS research results so that operational and perhaps commercial users can acquire the technology and use it to solve their own security problems. However, commercial vendors must be able to make a business case for investing their own resources to commercialize the technology. Given the slow acceptance of MLS technology by the DoD, vendors' reluctance to venture into this market is understandable. It makes our technology transfer efforts more difficult as well as more important.

Before research results can be commercialized, we must be able to present a computing and

*Consultant to NRL

operational use approach that can both convince businesses to risk their capital and convince operational users that MLS technology will allow them to be more successful in performing their jobs. Even when we understand what a system must do, we don't always know how to use MLS technology both to support the user in doing his job and to ensure that the information managed by the system is protected. We need a secure system engineering approach as well as valid research results. We will identify some areas of current interest and describe how the SINTRA approach can be used to solve these problems.

Additionally, we should be sensitive to the following observations that were recently published in the Joint Security Commission's (JSC) report [14]:

> *Those who steadfastly resist connectivity will be perceived as unresponsive and will ultimately be considered as offering little value to their customers.*
>
> *Our paradigm for managing information security must also shift from developing security for each individual application, system, and network to developing security for subscribers within the worldwide utility.*

## 2 Technology Transfer

We first identify some questions that should focus technology transfer efforts:

- What are our technology transfer goals?
- To whom do we wish to transfer our technology?
- Are we really transferring technology or are we instead providing research results upon which technology can be built?
- Do we expect someone else to figure out how the research results can be used? How should they be used?
- Do they support distribution of data?
- How difficult is it to integrate new technology into this security paradigm?
- How hard is it to reconfigure computing resources to accommodate change in the operational environment?

Our primary technology transfer goal is to provide a technical capability that allows secure access to all the information a user needs to do his job, no matter whether that user is a DoD, government, or commercial worker and no matter where a user or the data he needs are located. The challenge for database security research and for technology transfer efforts is to develop systems that distribute data securely and reliably.

Who is our customer? Our first concern has to be the DoD customer. In this respect, our job is perhaps more difficult. Computing resources for operational users are only beginning to be upgraded to modern, state-of-the-art processors. Although client-server architectures and relational database technology have been introduced, the re-engineering of operational application software has not exploited the full capabilities of the technology.

## 3 Legacy Systems

Today, tactical data is communicated through formatted military messages concerning readiness, schedules, equipment failure and its impact on mission readiness, location of friendly forces, commercial traffic, enemy platforms, and more. Users must access several systems, sometimes at different security levels, to retrieve information through the application running on each system. These systems were developed to support a specific mission and are updated through the parsing of a formatted message. The "data fusion" problem is, in part, due to heterogeneous representation schemes, inclusion of collection- specific information with the data, data access through applications, and little support for interoperability with other information systems. Changes made to related data should result in consistent information. However, updates made to a single application as a result of one formatted message are not necessarily propagated to other applications.

A major challenge for DoD is to move away from application-specific, "stove-pipe" information processing to cooperative, distributed computing architectures. "Stove-pipe" refers to systems that have been designed to accept data, transform it, and provide the output for a specific organizational objective without any capability for interoperating with other systems. Often these systems have been modified to support a changed operational mission. These systems are quite fragile, difficult to modify, and expensive to maintain. Stove-pipe systems can not easily migrate to newer technology because no one really understands what the system does. Yet, organizations depend on these systems for their corporate survival and are reluctant to risk change in case some valuable corporrate capability will be lost.

Today, organizations understand that both their information processing needs as well as information technology itself will change. This realization imposes additional requirements to develop systems that can take advantage of new technology to accommodate changing organizational requirements.

Recent trends in cooperative, distributed computing promote use of powerful client workstations for application-specific processing, like situation assessment and mission planning, and use of data management servers to provide transparent, reliable access to consistent data. When new technologies, such as relational database management systems, are introduced, we must present strategies for the migration of legacy systems upon which operational users depend to new operational capabilities that allow them to do their jobs more efficiently and effectively. Likewise, when MLS technology is introduced, a graceful migration strategy must be included and the MLS technology had better support a reasonable, pragmatic approach for support of distributed, cooperative computing. Recent publications, [15] and [16], describe in painful detail efforts to reengineer legacy systems to take advantage of newer technology and support more comprehensive enterprise computing requirements.

System development for a distributed computing environment is quite different than for an autonomous, stand-alone system. It is more closely analogous to development for parallel processors. Hence, the technical concerns are different. Likewise, security engineering for a distributed computing environment is different. As noted earlier from the JSC Report [14], we must work on solutions for MLS distributed computing and move away from our current focus on composition of MLS systems. To achieve this goal, our research efforts must concentrate on security architectures for cooperative, distributed computing environments, not security architectures for stand-alone, MLS systems. In such environments, we can take advantage of physical separation to achieve our security goals and provide reliable, transparent access to the data users need.

## 4 The SINTRA Approach

The philosophy of protection for SINTRA was strongly influenced by a desire to minimize the amount of software, both trusted and untrusted, which runs on a high assurance, painstakingly crafted TCB and, at the same time, to maximize the database capability without introducing security vulnerabilities. These objectives resulted in a protection mechanism that not only mediates access to data but also mediates access to general computing execution cycles because protection critical execution cycles are separated from general execution cycles. This approach has produced an extremely strong protection mechanism that is not susceptible to the kinds of vulnerabilities that are inherent in conventional MLS operating system approaches to system security, including those for data management systems. Because security and application execution cycles are not shared, malicious code in an untrusted application cannot exploit covert channels in the TCB. In other words, SINTRA limits the opportunities for exploitation. Replication ensures that information only flows upward in a security lattice.

SINTRA provides a straightforward, understandable approach to distribution and autonomy. Physical separation protects information. Strong, effective identification and authentication (I&A) mechanisms must be available on every component in a distributed system. Data need be replicated only when the data will be shared. In the SINTRA approach to data management, replica control is the only MLS requirement. Distribution requirements can be addressed at a single security level and a trusted replica controller provides the capability for sharing the information at different security levels. If the components in a distribution are heterogeneous, the translation mediation can be handled at a single level. When a data owner enters into an agreement to share information with another user, that data and changes to it can simply be replicated and sent to a cooperating computing resource accessible to the other user. In general, replication promotes the availability and sharing of information since the data owner is not required to expend his processing cycles to support retrievals by other users. The management of and access to the data copy, including any semantic translation requirements, can be provided securely, reliably, and transparently to the requesting user through resources available to him.

Current untrusted data management technology has turned to data replication to ensure data availability, reliability, autonomy, and fault tolerance. Replication provides a migration path for legacy systems to reengineered systems. First, if a transaction can be defined for a legacy system, updates to that system can be replicated to the reengineered systems with suitable translations required by the new technology or by a semantic difference in the data representation. The reengineered system can be operated alongside the legacy system until we have confidence that the legacy system's required capabilities have been successfully implemented in the reengineered system.

While SINTRA does not provide any greater support for the reengineering of legacy systems than any MLS relational database or other data management technology, it does provide a graceful migration opportunity for the new technology to be inserted. The SINTRA approach for legacy systems requires a clear specification of what an update and what a transaction are for the legacy system. These updates can be replicated to legacy systems running at higher security levels. This confederation of legacy systems connected through a replication controller provides an MLS capability for the legacy system.

The SINTRA architecture easily accommodates technology upgrades to any relational database. While the particular implementations of transaction management for other technologies, such as object oriented DBMSs, have to be built, we believe that an MLS capability can be developed for these technologies with an affordable security overhead.

In a cooperative, distributed computing environment, a trusted replica controller becomes a building block for secure systems engineering. In effect, security can be integrated without agonizing about whether some feature of the desired system is protection critical because the protection mechanisms are completely separated from the system's conventional features. The SINTRA architecture does not allow illegal information flows. The trusted replica controller is analogous to a cryto unit in the COMSEC world.

Probably most important to the operational community, however, the SINTRA approach allows MLS technology to exploit current advances in information technology. The use of trusted replica controllers as MLS connectors allows users and developers to concentrate their development resources on implementing systems that satisfy operational requirements rather than focusing on providing a MLS capability. The lifecycle management of single-level systems is more affordable than lifecycle management for the MLS distribution of MLS systems as well.

## 5 Conclusions

We believe that this architectural approach makes reasoning about security in the large possible and permits the scaling up of formal modeling, specification, and proof technology. Separation of the protection critical execution cycles from general processing cycles makes this advantage possible. Hence, well-understood assurance techniques can evolve incrementally to provide assurance for future cooperative, distributed computing solutions.

With these advantages, how have our technology transfer efforts faired? We have had some hard-won successes. However, like the vision for cooperative, distributed computing, we need a "killer" application to demonstrate the technology. Both users and program managers have difficulty understanding MLS in the large and are not always able to make the paradigm shift from stand- alone MLS systems to an MLS confederation of systems. They are reluctant to accept MLS data management without a complete security engineering solution for all their security problems. Issues that must be addressed are distributed identification and authentications (I&A), a user capability to write at different security levels, key management, and more.

## References

1. Froscher, J.N., and C.L. Meadows, "Achieving a Trusted Database Management System Using Parallelism," in Database Security II: Status and Prospects, C.E. Landwehr, ed., North Holland, 1989, pp. 151-160.

2. Costich, O., and I. Moskowitz, "Analysis of a Storage Channel in the Two Phase Commit Protocol", Proc. of the Foundations of Computer Security Workshop IV, Franconia, NH June 1991, pp. 201-208.

3. McDermott, J., S. Jajodia, and R. Sandhu, "A Single-Level Scheduler for the Replicated Architecture for Multilevel-Secure Databases", Proc. 7th Annual Computer Security Applications Conference, San Antonio, December 1991, pp. 2-11.

4. Costich, O. "Transaction Processing Using an Untrusted Scheduler in a Multilevel Database with Replicated Architecture", in Database Security V: Status and Prospects, eds. S. Jajodia and C. Landwehr, North-Holland, 1992, pp. 173-190.

5. Costich, O. and J. McDermott, "A Multilevel Transaction Problem for Multilevel Secure Database Systems and Its Solution for the Replicated Architecture", Proc. 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, California, May 1992, pp. 192-203.

6. Kang, M., H.G. Dietz, and B. Bhargava, "Data Dependence Analysis for an Untrusted Transaction Manager in a Multilevel Database System" Proc. of ISMM First International Conference on Information and Knowledge Management, Baltimore, 1992, pp. 441-448.

7. McDermott, J. and S. Jajodia, "Orange Locking: Channel-Free Database Concurrency Control via Locking", presented at 6th IFIP Working Conference on Database Security, August 1992, Vancouver, British Columbia, pp. 271-288.

8. Kang, M., J.N. Froscher, and O. Costich, "A Practical Transaction Model and Untrusted Transaction Manager for a Multilevel-Secure Database system" Proc. 6th IFIP Working Conference on Database Security, August 1992, Vancouver, British Columbia, pp. 289-310.

9. Kang, M., and I. Moskowitz, "A Pump for Rapid, Reliable, Secure Communication," Proc. 1st ACM Conf. on Computer and Communications Security, Fairfax, VA, Nov., 1993, pp. 119-129.

10. Kang, M. H., and R. Peyton, "Design Documentation for the SINTRA Global Scheduler," NRL Memorandum Report #5542-93-7362, June 30, 1993.

11. McDermott, J., and R. Mukkamala, "Performance analysis of transaction management algorithms for the Sintra replicated- architecture database system," Proc. Seventh Annual IFIP WG11.3 Working Conference on Database Security, Huntsville, AL, Sept. 1993, pp. 216-240.

12. Costich, O., and M. Kang, "Maintaining multilevel transaction atomicity in MLS database systems with replicated architecture," Proc. Seventh Annual IFIP WG11.3 Working Conference on Database Security, Huntsville, AL, Sept. 1993, pp. 333-357.

13. Kang, M. H., Costich, O., and Froscher, J. N. Using object modeling techniques to design MLS data models. The OOPSLA Conference Workshop on Security in Object-Oriented Systems (1993).

14. U. S. Joint Security Commission, "Redefining Security: A Report to the Secretary of Defense and the Director of Central Intelligence, Washington, DC, 28 February 1994.

15. P. Aikens, A. Muntz, and R. Richards, "DoD legacy systems: reverse engineering data requirements," CACM, Vol. 37, No. 5, pp.26-41, May 1994.

16. M. Brodie, "The promise of distributed computing and the challenges of legacy information systems", Advanced Database Systems: Proceedings of the 10th British National Conference on Databases, P.M.D. Gray and R. J. Lucas (eds.), Springer-Verlog, New York/Heidelburg, 1992.

## Discussion: Technology Transition

**Discussion Leader: Tom Haigh, Secure Computing Corporation**

(paper not available)

## INFERENCE CONTROL

# Inference and Knowledge Discovery

Donald G. Marks

Department of Defense, Office of INFOSEC, computer science
Ft. Meade, Md.

*Abstract: Inference control has become a topic of considerable interest in secure database implementation. It is generally recognized that access to certain types of information enables the user to infer other information, even some that should not be available to them. This generally occurs because users are able to construct datasets in ways that were not anticipated by the system designer. Knowledge Discovery techniques are designed to automate this process and therefore pose an inference threat. Current inference control techniques are inadequate to protect against this threat since they are limited to dealing with rules expressed at the schema level, that is, functional dependencies. Knowledge Discovery, however, may also be used by the system designer to find rules relating low and high data. If all these rules are classified "High" then there can be no successful inference attacks against the knowledge in the database.*

## 1.0 Definition of Inference

Inference control has become a topic of considerable interest in secure database implementation. It is generally recognized that access to certain types of information enables the user to *infer* other information, even some that should not be available to them. Such inference does not take place magically, rather it is the integration of techniques applicable to databases and those utilized by humans in making abstractions.

As a general rule, *inference* control is concerned with protecting *knowledge*, not data. Individual data items are properly protected by standard classification techniques. Knowledge, however, is inferred from a quantity of data, or a set of data associated with attributes. In this study, it is assumed that the data is stored in a relational database consisting of a series of tables. Each table represents one type of entity, with the column labels identifying the attributes, or properties, of the entities. Each row in a table represents a specific instance of an entity associated with that table and is identified by a unique primary key. In a secure database context, preventing knowledge from being released requires preventing the release of both the data and the attributes in a manner where they can be associated into a sensitive conclusion. The numbers and/or letters in a database are meaningless until they are associated with an attribute. For example, the word "Washington" could be a person's name, a city, a state, or a codeword. Numbers are even less meaningful without knowing the applicable attribute. Data are only meaningful when assigned to an attribute, or set of attributes. The ability to determine these attributes associated with the data, is the critical point of inference. Inference in a database is said to occur if, by retrieving a set of tuples {T}, having attributes {A}, from the database, it is possible to specify a set of tuples {T'}, having attributes {A'}, where $\{T'\} \neq \{T\}$ or $\{A'\} \neq \{A\}$.

*Definition of Database Inference* (1): ({T},{A}) implies ({T'},{A'}) if there exists a rule, *R*, that transforms ({T},{A}) into ({T'},{A'}).

Equivalent methods of stating this are:(1) IF ({T}, {A}) THEN ({T'}, {A'}); or

(2) ({T},{A})⇒({T'},{A'}).

In Figure 1, knowledge of the (low classified) data set, B, *implies* knowledge of the (high classified) data set, A. Such inference capability requires a rule, R1, which the low cleared user either knows or can read. If such a rule exists, then the data set B must also be classified high. In general this principle is well known and inference is considered whenever data is initially classified.

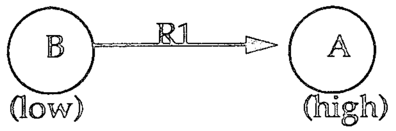


Figure 1.

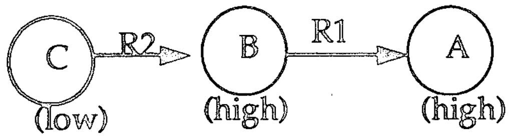


Figure 2

If a dataset B is re-classified as high in order to prevent some inference threat, the entire database may need to be re-examined to determine if there exists another set C, and a rule R2, such that C implies B, as shown in Figure 2. If such a dataset exists, then it must also be re-classified high, in order to prevent a low user from following a chain of inferences, eventually leading to the high data, A. The classification official is tasked with determining all such chains, regardless of their complexity. Note, however, if, at any point, the link connecting the low and high datasets is only connected by a high classified rule (so the low cleared user could not possibly know of the connection between the data), then the rest of the chain need not be re-classified high.

However, it is possible, especially for complicated databases, for some of the data relationships to be unanticipated by the classification official. In this case, the malicious user may be able to construct a set of data in some way not envisioned by the classification official. This dataset may be combined with a known rule in order to infer high classified data.

Now consider data outside the database, that is, "common knowledge" data. If such common knowledge data (outside the database) can imply high classified data (inside the database), through common knowledge rules, then the high data cannot be protected by the database. It is likewise not reasonable to expect a DBMS to protect data outside the database. That is, the database will not know if low classified data can be used to infer high classified data, if that high data is not in the database. The problems involving data outside the database have always existed, and they are neither reduced nor magnified by putting the data in a computer system. At best, a database can prevent low data inside the database from being used to compromise high data that is also inside the database, that is, it avoids *self-compromise*. Therefore, the most optimistic situation may be stated as:

*If all the tuples in {T} and {T'} are in the database, and all the properties in {A} and {A'} are attributes in the database, then ({T},{A}) $\Rightarrow$ ({T'},{A'}) is an inference rule capable of being controlled by the database.*

## 2.0 Current Efforts

Inference control efforts to date have dealt with the model of data and rules described in Section 1 starting with the definitive formal model presented by Morgenstern [MORG87]. Practical efforts have approached the problem either by: 1) finding low classified datasets that imply high classified datasets using only the known functional dependencies as rules; or 2) finding new rules relating data and checking the rule interactions for inference.

Approach (1) is taken by Binns [BINN92], who does not assume any database structure, and considers arbitrary dataset combinations formed from the schema. The computational complexity of such an approach limits its usefulness. Garvy et. al. [GARV92], assume a highly structured database schema, and construct datasets with those dependencies that are known in advance. While this approach is computationally tractable, it fails to address many inference rules, especially the more subtle ones. Both of these approaches are limited to functional dependencies as specified in the database schema.

Approach (2) is taken by Hinke [HINK92], who does not derive the rules from the schema but has developed a knowledge engineering tool to assist the designer in defining inference rules. Thuraisingham [THUR91] also uses a knowledge engineering approach, but with conceptual graphs as a representation. These approaches extend the capability of the database, but are incapable of determining when all the rules have been found. They offer no possibility for assurance that all inference rules have been considered. Other studies have tended to focus upon sub-classes of the problem, or specific types of inference problems and solutions.

In either case, the solution is to classify additional sets of data as high. As the data was initially classified, it was scrutinized to insure that it cannot be easily derived from unclassified information. However, we have now started re-classifying data in the database for the purpose of controlling inference. We have no assurance that such data has received adequate scrutiny regarding outside influence. It is desirable for the inference controller to apply this scrutiny rather than to refer the problem back to the classification officer. This iterative solution is a feature of most of the above proposals.

The major problem with these approaches is the requirement to specify all the possible inference rules, and then check their interactions among themselves and with the low classified data sets. Some of the aforementioned studies even attempt to specify rules utilizing data outside the database. It is suggested that this is the wrong approach for a system designer or administrator. The low data/rule approach is basically the approach taken by a system attacker, but system designers or administrators have more information, namely access to the high data. This information may be utilized if we strengthen our definition of inference to be:

*Definition of Database Inference (2)*: ({T},{A}) implies ({T'},{A'}) if, *and only if*, there exists a rule, *R*, that transforms ({T},{A}) into ({T'},{A'}).

This gives us a method to provide some assurance regarding inference control. *The datasets may be used to determine the rules.* Since the number of possible datasets is finite, the number of possible rules is finite, and the process will eventually terminate. These rules are found by using artificial intelligence (AI) techniques, basically *knowledge discovery* methods. Such AI programs may be assured to find all the relevant rules, but only those rules, linking high and low data.

## 3.0 Knowledge Discovery

What is knowledge discovery? Knowledge discovery (KD) is the process of finding knowledge, or meaning, from seemingly unconnected data. Security practitioners should immediately be concerned by this definition. Here is a body of science whose purpose is to find inference rules between the data. No longer are we dealing with a person, who might overlook certain meaningful data relationships, or simply be incapable of checking them all. Rather, we have a computer program which, given enough cpu cycles, is guaranteed to find everything. Therefore, if a low classified chain of reasoning exists, leading eventually to high data, we can be certain that it will be found. We can no longer assume that it is unlikely that a person would know all the relevant inference rules: all such rules will be found.

Knowledge discovery deals with the database extension, that is, the actual data instances populating the database schema. KD techniques can therefore handle partial inferences, element level labeling, and content-dependant inference rules. Dealing with the actual data does require, however, that all inference rules expressible in the database schema are reflected in the actual database instances. In this case, the KD derived rules will be a superset of those special-case rules handled in the previous studies. Of course, such flexibility and completeness does not come without a price: KD programs are very cpu intensive. However, with the continued decline in the cost of computing power, such programs are becoming more reasonable every day. System attackers frequently have the luxury of dedicating their system, for days on end, to the task of compromising the attacked system's data.

One interesting feature of KD techniques is their utilization of *material implication* rules. Such rules are not based upon causality, but upon simple concurrent satisfaction of predicates. Material implications may be expressed as IF ... THEN... statements. Therefore, if there exists a chain of inference, $A_1$ *implies* $A_2$ *implies* ... *implies* $A_n$, then we may shorten this chain to the endpoints, or $A_1$ *implies* $A_n$. A KD algorithm will find this rule without bothering to actually follow the chain. Whether or not the datasets, $A_2 ... A_{n-1}$, are inside or outside the database has no effect on the inference discovery. The existing projects that are attempting to actually place outside knowledge into the database, may therefore be making the problem harder than necessary.

## 4.0 Conclusions

If we desire any sort of assurance that a system is protected against inference attacks, we cannot rely upon the techniques developed up until this time. None of them are suitable for protection against KD types of attack. Perhaps the only method capable of providing real protection against a KD-type attack is to use the same KD techniques to implement security. Here we have an advantage over the attacker. An attacker can only access low-classified data, while a trusted process can access both high and low data. The low cleared KD tool must do a great deal of searching of the low data in order to locate appropriate rules and carefully construct the target dataset. In a worst case, all possible combinations of low classified datasets would have to be compared. System security only requires that *rules* connecting low data to high data be unknown to the low system users. Hence the trusted KD tool need check all combinations of low data and high data, a formidable task, but usually much easier than checking all combinations of datasets at the same level. The rules found by this method will contain all true inference rules, plus many false rules that appear true due to coincidence. Each rule must, of course, be evaluated before it is included in the system.

## 5.0 References

[BINN92] Binns, L., August 1992, "Inference through secondary path analysis," *Proc. of the 6th IFIP Working Conference in Database Security,* Vancouver, BC.

[MORG87] Morgenstern, M., "Security and Inference in Multilevel Database and Knowledge Base Systems". *Proc. of the ACM SIGMOD Conference,* San Francisco, April, 1987.

[THUR91] Thuraisingham, B., "The use of conceptual structures for handling the inference problem", *Proc. 5th IFIP Working Conference on Database Security,* Shepherdstown, WV., Nov. 1991.

[HINK92] Hinke, T.N., and H. Delugach, "Aerie: an inference modeling and detection approach for databases", *Proc. 6th IFIP Working Conference on Database Security,* Vancouver, B.C., Aug. 1992.

[GARV92] Garvey, T.D.,T.F. Lunt, X. Qian, and M. Stickle, "Toward a tool to detect and eliminate inference problems in the design of multilevel databases", *Proc. 6th IFIP Working Conference on Database Security,* Vancouver, B.C., Aug. 1992.

# Inference Control Tool: Project Summary*

Thomas D. Garvey
Artificial Intelligence Center
SRI International
Menlo Park, California 94025

## 1 Introduction

The advent of commercially available trusted database systems introduces the capability to manage data at a variety of sensitivities and to enforce security policies that prohibit the unauthorized disclosure of information to unauthorized or insufficiently authorized individuals. With these products, data are labeled with their degree of sensitivity and protected accordingly. However, these products cannot protect data that is incorrectly labeled. One difficulty is that highly sensitive data may be inferred from data labeled lower[1]. In such cases an *inference problem* exists. An inferential link that may allow highly sensitive information to flow to a low user is termed an *inference channel* [?, ?]. It is the difficult task of the data designer to label the data so that the labels accurately reflect the actual sensitivity of the data and adequately protect the information from inference. The latter aim is extremely difficult for the human data designer to attain. SRI has developed an automated tool that can identify potential inference channels in a labeled database. DISSECT [?, ?, ?] (Database Inference System Security Tool) can be used interactively by a data designer to analyze candidate database schemas to assist in the detection and elimination of inconsistent labeling that can constitute inference problems. DISSECT uses *schema-level analysis* to avoid the costly task of data-level analysis with every database query.

DISSECT can detect both compositional inference channels and inference channels that involve type-overlap and near-key relationships. A potential *compositional inference channel* exists if two attributes are connected by a pair of paths consisting of composed foreign key relationships, where the two paths may have different sensitivities. A relationship can be inferred between any pair of entities that are connected by a sequence of foreign key relationships. If a table contains a foreign key to a second table, then there is a functional relationship from entities described by the first table to entities described by the second. A foreign key relationship from the second table to a third implicitly defines a *composed* functional relationship from entities described by the first table to entities described by the third. If there is another sequence of foreign key relationships connecting the first and third tables, and accessing the two sequences may require different authorizations, there may be a compositional channel, since the two sequences of foreign key relationships may describe the same or a too closely related relationship between the first and third entities.

*This research was supported by the United States Air Force, Rome Laboratory, and the Advanced Research Projects Agency under Contract F30602-91-C-0092.

[1]We use the terms "high" and "low" informally to refer to data that is more or less sensitive.

Compositional channels involve relationships that are explicitly defined in the database schema. The foreign key relationships that compose them are mappings from an attribute[2] of one relation to the primary key of another. The schema contains the information required to search for compositional channels, but the security of the database can still be compromised by more indirect methods. A foreign key relationship requires that the second attribute be a primary key and that every value of the first attribute be included among the values of the second. Foreign key relationships specify the join operations that the data designer intends the database user to perform. However, a user can join any pair of attributes that have values in common. Moreover, neither attribute need be a primary key. If one is a *near key*, joining on it can yield information about dependent attributes nearly as well as the primary key. DISSECT allows the data designer to declare information about attribute joinability and near keys to enable detection and elimination of the additional inference channels they allow.

Rather than require that the data designer state explicitly list every pair of attributes that are joinable, we allow him to associate *types* with attributes. Attributes whose types overlap are joinable. A *type-overlap* relationship occurs between two attributes when the two attributes have been declared to be of the same type and also have some overlap in the allowed sensitivity labels for data elements of that type. For example, there may be some overlap between attributes home-phone-number and office-phone-number, if they are both declared to be of type phone-number, and if elements of each may also match in sensitivity level. Intuitively, a type-overlap relationship is one which would allow the two attributes to be joined on matching data values and sensitivities. A potential inference problem exists if there is a pair of different-sensitivity paths between the same two entities, where the high path consists of a sequence of foreign key links, and the low path consists of both foreign key and type-overlap links. Intuitively, we are looking for ways a low user could use both declared foreign key relationships and fortuitous type-overlap relationships to compromise an explicit high relationship consisting of a sequence of one or more foreign key relationships. To allow DISSECT to discover inference channels that involve type-overlap relationships, the data designer must make type declarations for the attributes in the database. Inclusion of type-overlap relationships in DISSECT's detection algorithms allows DISSECT to detect inference problems caused by a user's *ad hoc* queries that the data designer might not have considered.

The detection of inference channels that involve type-overlap and near-key relationships require the data designer to make type declarations for the attributes in the database. The type declarations need not be complete; where the data designer has not made type declarations, DISSECT assumes nonoverlapping types.

In related work [?], Binns considered two attributes to be related if they had the same name. He created inference paths by concatenating such relationships. A potential problem was detected as a pair of such paths connecting the same end entities but having different security levels. Some problems with his approach are that (1) many spurious inference problems will be detected, since two attributes are not necessarily related or even joinable simply because they have the same name (his solution to this was to impose the unrealistic requirement that attribute names be unique across the database), and that (2) many relationships that could contribute to inference paths could go undetected, since attributes can be meaningfully joined even though they do not share the same name. Our type-overlap approach achieves the intent of Binns' approach (namely, of detecting problems that could not have been anticipated by the data designer), but will detect all and only

[2]For simplicity, we will discuss here only the case of relations among single attributes and not primary or foreign keys composed of multiple attributes.

those paths formed of meaningful relationships.

## References

[1] T.D. Garvey, T.F. Lunt, and M.E. Stickel. Characterizing and reasoning about inference channels. *Proceedings of the Fourth RADC Workshop on Database Security,* Little Compton, Rhode Island, April 1991.

[2] T.D. Garvey, T.F. Lunt, and M.E. Stickel. Abductive and approximate reasoning models for characterizing inference channels. *Proceedings of the Fourth Workshop on the Foundations of Computer Security,* Franconia, New Hampshire, June 1991.

[3] X. Qian, M.E. Stickel, P.D. Karp, T.F. Lunt, and T.D. Garvey. Detection and elimination of inference channels in multilevel relational databases. *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy,* Oakland, California, May 1993.

[4] M.E. Stickel, X. Qian, T.F. Lunt, and T.D. Garvey. *Inference Channel Detection and Elimination (Second Interim Report)*, Computer Science Laboratory, SRI International, Menlo Park, California September, 1993.

[5] M.E. Stickel. Elimination of inference channels by optimal upgrading. *Proccedings of the 1994 IEEE Symposium on Research in Security and Privacy,* Oakland, California, May 1994.

[6] L.J. Binns. Inference through secondary path analysis. *Proceedings of the Sixth IFIP Working Conference on Database Security,* August, 1992.

## Discussion: Inference

**Discussion Leader: Teresa Lunt, SRI International**

(paper not available)

# USABILITY OF MULTILEVEL SECURE DATABASE MANAGEMENT SYSTEMS

# DESIGN AND IMPLEMENTATION OF MULTILEVEL DATABASES

*Ravi Sandhu*

Department of Information and
Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444
`sandhu@isse.gmu.edu`

This paper briefly describes ongoing research at GMU on the problem of designing and implementing multilevel databases. In a nutshell the objective of our research is to close the semantic gap between sophisticated requirements of MLS applications and the relatively meager facilities provided by emerging MLS DBMS products. There is a missing links in previous research in the MLS database arena. Previous research has tended to focus either on

- the behavior of a relational MLS DBMS and problems associated with implementing this behavior in different MLS architectures, or
- on stating requirements for an MLS database using semantic data models and related techniques.

There are several notable exceptions to this statement. Sell and Thuraisingham [ST94] have recently proposed a Multilevel Object Modeling Technique (MOMT), patterned on OMT, for designing multilevel database applications using a relational MLS DBMS platform. Lewis and Wiseman [LW93] have also recently described a case study in mapping requirements stated in the SPEAR notation into SWORD and SeaView. The RADC workshop several years ago did a case study of mapping requirements into systems [Smi89, ST89, Hin89, Mai89, Stu89]. The TTCP XTP-1 Workshop on Research Progress in MLS Relational Database held prior to the 1994 RADC workshop also poses a case study.

Our research seeks to reconcile these two streams of activity by addressing the missing-link question of how to achieve the stated requirements on a given data model of MLS relations. It will build upon the prior research cited above. The Sell-Thuraisingham and Lewis-Wiseman efforts were targeted at element-level labeling of MLS relations. Our project will go beyond this work by also considering tuple-level labeling. This is particularly important because emerging MLS DBMS products provide tuple-level labeling rather than the element-level labeling discussed in most of the research literature. This fact widens the semantic gap identified above, and makes the proposed research all the more topical and relevant to the practitioner seeking to build MLS applications on these emerging platforms.

The basic premise of the proposed research is that the theoretical expressive power of fairly simple models can be surprisingly general. The classic example of this is Turing machines, and related automata, which are capable of executing all computable activity. At the same time, simple models are usually not practical to use directly even though they are theoretically capable of solving the problem at hand. It is therefore necessary to develop additional tools to close the semantic gap in a practically useful manner, rather than just declaring theoretical adequacy of a simple model. This approach to closing the semantic gap has been repeatedly employed in computer systems. Our expectation is it will also succeed in the arena of MLS relational databases.

The reason for considering tuple-level labeling is that most of the emerging MLS DBMS products are adopting this approach. This is a natural approach for DBMS vendors, in that the tuple is the basic storage and retrieval unit in typical relational DBMS implementations.

There has been some theoretical discussion of equivalence between tuple-level labeling and element-level labeling. Qian and Lunt [QL93] have published an interesting claim that tuple-level labeling is equivalent to the SeaView model (under a particular definition of equivalence) . We feel this issue needs to be studied more carefully, and in a broader context than the SeaView model. The notion of what is meant by equivalence itself needs a rigorous examination. We now illustrate the subtleties involved here by contrasting two interpretations of tuple-level labeling.

## A SIMPLISTIC INTERPRETATION OF TUPLE-LEVEL LABELS

Let us consider a simple mapping from tuple-level labels to element-level labels. Say we have the following tuple

$$(a_1, a_2, ..., a_n, c)$$

where the $a_i$'s are the individual data elements of the tuple, and $c$ is the security label on the tuple. The simplest mapping to element-level labelling is to simply put $c$ as the label of each of the individual elements, as well as let $c$ be the tuple class. This would give us the following tuple (with element-level labeling).

$$(a_1, c, a_2, c, ..., a_n, c_n, c)$$

Each $c$ labels the data element to its left, except for the rightmost one which labels the entire tuple. This simple mapping severely cripples the expressive power of tuple-level labeling. It is impossible to translate the following tuple with element-level labels to an equivalent one with tuple-level labels.

$$(a_1, U, a_2, S, ..., a_n, S, S)$$

This tuple associates an unclassified data element $a_1$ with a number of secret data elements $a_2 \ldots a_n$. Such an association cannot be expressed with this simplistic interpretation of tuple-level labeling. But this is not the only possible interpretation.

## AN ALTERNATE INTERPRETATION OF TUPLE-LEVEL LABELS

Let us consider an alternate interpretation. We caution the reader that this interpretation is being presented only for sake of example. We are not suggesting it as an interpretation to be recommended. Finding useful interpretations of tuple-level labeling is a task for the proposed research.

Let us assume that $A_1$ (i.e., the first attribute) of a tuple is the apparent key. Now suppose the following tuples are coexisting in a relation (with tuple-level labeling).

$$(a_1, a_2, ..., a_n, U)$$

$$(a_1, a_2', ..., a_n', S)$$

Note that there are two tuples with the same apparent key value (i.e., $a_1$), so this is a form of polyinstantiation. Now consider the following mapping to element-level labeling.

- Data elements outside of the apparent key inherit the label of the tuple.
- The apparent key is assigned a label equal to the greatest lower bound of the labels of all tuples in which it occurs.

For the pair of tuples shown above we obtain the following two tuples (with element-level labeling) respectively.

$$(a_1, U, a_2, U, ..., U, a_n, U, U)$$

$$(a_1, U, a_2', S, ..., a_n', S, S)$$

Furthermore, consider the following tuple (with element-level labels) which we could not map to an equivalent one with tuple-level labels under the previous interpretation.

$$(a_1, U, a_2, S, ..., a_n, S, S)$$

With the current interpretation we can attempt to translate this into two tuples (with tuple-level labels) as follows.

$$(a_1, null, ..., null, U)$$

$$(a_1, a_2, ..., a_n, S)$$

The first of these essentially fixes the label of $a_1$ at $U$. The second gives the $S$ values associated with $a_1$. Moreover, these two tuples can be interpreted as respectively corresponding to the $U$ and $S$ views of the tuple (with element-level labels) they were derived from.

This alternate interpretation illustrates the important point that it is possible to have a richer semantics for tuple-level labeling than obtained by the simplistic interpretation given earlier. There a number of research questions that need to be addressed here.

Firstly, it is not clear if there is a consistent and useful semantics for tuple-level labeling along the lines sketched out above. The work of Qian and Lunt address this question from one perspective. We feel that a more comprehensive study of this problem is called for, particularly since considerable progress on understanding polyinstantiation has been made in the meantime. Moreover, Qian and Lunt do not consider dynamic aspects of the relations, such as update semantics. In short, much work remains to be done.

Secondly, we must consider the semantics of tuple-level labeling supported in the emerging products. In particular, the update behavior is determined by these DBMS's. It therefore constrains the range of interpretations we can impose on these products.

Thirdly, we must consider how to practically map element-level requirements to tuple-level models. Even if we can establish some kind of theoretical equivalence, we will still need tools and possibly human guidance in achieving an effective mapping.

## References

[Hin89] Thomas H. Hinke. Secure database design panel. In *Fourth Annual Computer Security Application Conference*, page 323, Tucson, AZ, December 1989.

[LW93] Sharon Lewis and Simon Wiseman. Database design & MLS DBMSs: An unhappy alliance? In *Ninth Annual Computer Security Application Conference*, pages 232–243, Orlando, FL, December 1993.

[Mai89] Bill Maimone. Oracle corporation homework problem solution. In *Fourth Annual Computer Security Application Conference*, page 324, Tucson, AZ, December 1989.

[QL93] Xiaolei Qian and Teresa Lunt. Tuple-level vs. element-level classification. In B. Thuraisingham and C.E. Landwehr, editors, *Database Security VI: Status and Prospects*, pages 301–315. North-Holland, 1993.

[Smi89] Gary W. Smith. Multilevel secure database design: A practical application. In *Fourth Annual Computer Security Application Conference*, pages 314–321, Tucson, AZ, December 1989.

[ST89] Paul Stachour and Dan Thomsen. A summary of the ldv solution to the homework problem. In *Fourth Annual Computer Security Application Conference*, page 322, Tucson, AZ, December 1989.

[ST94] Peter J. Sell and Bhavani M. Thuraisingham. Applying omt for designing multilevel database applications. In T. Keefe and C.E. Landwehr, editors, *Database Security VII: Status and Prospects*, pages 41–64. North-Holland, 1994.

[Stu89] Edward D. Sturms. Secure database design: An implementation using a secure dbms. In *Fourth Annual Computer Security Application Conference*, page 325, Tucson, AZ, December 1989.

# DESIGN AND IMPLEMENTATION OF MULTILEVEL DATABASES

Ravi Sandhu
George Mason University
Fairfax, Virginia

## POLYINSTANTIATION, COVER STORIES AND CONFUSION

- Polyinstantiation can cause confusion if not managed properly
- Polyinstantiation need not cause confusion if managed properly
- Polyinstantiation can be completely eliminated if so desired
- Polyinstantiation can be used for cover stories if so desired

NOTE: Analogy to GOTO's in programming languages

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise U | Exploration U | Talos U | U |

UNCLASSIFIED INSTANCE OF SOD

| No. | Starship | Objective | Destination | TC |
|---|---|---|---|---|
| 1 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Spying S | Talos U | S |
| 2 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Exploration U | Rigel S | S |
| 3 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Spying S | Rigel S | S |

COVER STORIES

|  |  |  |  |  |
|---|---|---|---|---|
| 4 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Exploration U | Rigel S | S |
|  | Enterprise U | Spying S | Rigel S | S |
| 5 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Spying S | Talos U | S |
|  | Enterprise U | Spying S | Rigel S | S |
| 6 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Spying S | Talos U | S |
|  | Enterprise U | Exploration U | Rigel S | S |
| 7 | Enterprise U | Exploration U | Talos U | U |
|  | Enterprise U | Spying S | Talos U | S |
|  | Enterprise U | Exploration U | Rigel S | S |
|  | Enterprise U | Spying S | Rigel S | S |

CONFUSION

## CORE INTEGRITY PROPERTIES

- Entity Integrity
- Inter-Instance Integrity
- Polyinstantiation Integrity (PI)
- PI-tuple class

- Foreign Key Integrity
- Referential Integrity
- No Entity Polyinstantiation

## WHAT IS ALL THIS FUSS ABOUT?

- We cannot simply label data in a single-level system and thereby make it multi-level
- We cannot simply un-label data in a multi-level system and thereby make it single-level

| Starship | Objective | Destination |
|---|---|---|
| Enterprise | Exploration | Talos |

SINGLE LEVEL INSTANCE OF SOD

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |

ACCEPTABLE

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | S | Talos | U | S |

ACCEPTABLE

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | S | S |

ACCEPTABLE

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | S | Talos | S | S |

ACCEPTABLE

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | S | Exploration | S | Talos | S | S |

ACCEPTABLE

| Starship | Objective | Destination |
|---|---|---|
| Enterprise | Exploration | Talos |

SINGLE LEVEL INSTANCE OF SOD

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise S | Exploration U | Talos U | S |

UNACCEPTABLE

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise S | Exploration S | Talos U | S |

UNACCEPTABLE

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise S | Exploration U | Talos S | S |

UNACCEPTABLE

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise S | Exploration S | Talos S | S |

ACCEPTABLE

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise S | Exploration S | Talos TS | TS |

ACCEPTABLE

## ENTITY VERSUS ELEMENT POLYINSTANTIATION

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | S | Spying | S | Rigel | S | S |

ENTITY POLYINSTANTIATION

2 ENTITIES WITH THE SAME KEY

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | S | Rigel | S | S |

ELEMENT POLYINSTANTIATION

1 ENTITY WITH CONFLICTING INFORMATION ABOUT IT

## ENTITY VERSUS ELEMENT POLYINSTANTIATION

- **Entity polyinstantiation is incompatible with referential integrity.**
- **Entity polyinstantiation can be eliminated, but only by proactive mechanisms.**
- **Element polyinstantiation is useful for cover stories, if properly implemented.**
- **Element polyinstantiation can be easily prevented by reactive mechanisms.**

## ACCEPTABLE ELEMENT POLYINSTANTIATION

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | S | Rigel | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Exploration | U | Rigel | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | S | Talos | U | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Mining | C | Sirius | C | C |
| Enterprise | U | Spying | S | Rigel | S | S |
| Enterprise | U | Coup | TS | Orion | TS | TS |

# UNACCEPTABLE ELEMENT POLYINSTANTIATION

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | U | Rigel | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Exploration | U | Rigel | S | S |
| Enterprise | U | Spying | S | Talos | U | S |
| Enterprise | U | Spying | S | Rigel | S | S |

# ACCEPTABLE ELEMENT POLYINSTANTIATION

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Exploration | S | Talos | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Exploration | U | Talos | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Exploration | U | null | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Mining | C | Sirius | C | C |
| Enterprise | U | Exploration | U | Rigel | S | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Mining | C | Sirius | C | C |
| Enterprise | U | Exploration | U | Talos | U | S |

# REFERENTIAL INTEGRITY

| Starship | Objective | Destination |
|---|---|---|
| Enterprise | Exploration | Talos |

INSTANCE OF SOD

| Captain | Starship |
|---|---|
| Kirk | Enterprise |

PROPER REFERENCE FROM CS TO SOD

| Captain | Starship |
|---|---|
| Kirk | null |

NO REFERENCE FROM CS TO SOD

| Captain | Starship |
|---|---|
| Kirk | Battlestar |

DANGLING REFERENCE FROM CS TO SOD

## REFERENTIAL AMBIGUITY

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | S | Spying | S | Rigel | S | S |

| CAPTAIN | | SHIP | | TC |
|---|---|---|---|---|
| Kirk | U | Enterprise | S | S |

- Reference down introduces ambiguity due to entity polyinstantiation (Original SeaView)
- Reference at your level eliminates ambiguity (Revised SeaView)

## REFERENTIAL INCOMPLETENESS

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | S | Rigel | S | S |

| CAPTAIN | | SHIP | | TC |
|---|---|---|---|---|
| Kirk | U | Enterprise | S | S |

- **Reference at your level severely limits modelling power (Revised SeaView)**

## SOLUTION

- Eliminate entity polyinstantiation
- Otherwise choose between
  - referential ambiguity, OR
  - referential incompleteness

## ELIMINATION OF ENTITY POLYINSTANTIATION: PART I

- Preallocation of key space to security classes.
  - U Starships have names beginning with A-E
  - S Starships have names beginning with F-K
  - etcetera
- Keys assigned by a trusted user outside of the computer system.
- Keys assigned by a trusted subject in the computer system. Will introduce a low bandwidth covert channel.

# ELIMINATION OF ENTITY POLYINSTANTIATION: PART II

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Voyager | S | Spying | S | Rigel | S | S |

| CAPTAIN | | RANK | | TC |
|---|---|---|---|---|
| Kirk | U | Admiral | U | U |
| Spock | S | General | S | S |

REFERENCED RELATIONS

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | U | 15 | U | U |
| Kirk | Enterprise | U | 10 | S | S |
| Kirk | Voyager | S | 30 | S | S |
| Spock | Enterprise | S | 20 | S | S |
| Spock | Voyager | S | 15 | S | S |

REFERENCING RELATION

# ELIMINATION OF ENTITY POLYINSTANTIATION: PART II

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Voyager | S | Spying | S | Rigel | S | S |

| CAPTAIN | | RANK | | TC |
|---|---|---|---|---|
| Kirk | U | Admiral | U | U |
| Spock | S | General | S | S |

REFERENCED RELATIONS

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | U | 10 | S | S |

ACCEPTABLE REFERENCING RELATION

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | U | 15 | U | U |
| Kirk | Enterprise | U | 10 | S | S |

ACCEPTABLE REFERENCING RELATION

# ELIMINATION OF ENTITY POLYINSTANTIATION: PART II

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Voyager | S | Spying | S | Rigel | S | S |

| CAPTAIN | | RANK | | TC |
|---|---|---|---|---|
| Kirk | U | Admiral | U | U |
| Spock | S | General | S | S |

REFERENCED RELATIONS

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | S | 10 | S | S |

NO ENTITY POLYINSTANTIATION

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | U | 15 | U | U |
| Kirk | Enterprise | S | 10 | S | S |

ENTITY POLYINSTANTIATION!

## ELIMINATION OF ENTITY POLYINSTANTIATION: PART II

- Seems we cannot keep assignment of Kirk to Starship Secret without opening up possibility of entity polyinstantiation.
- We can convert the perceived entity polyinstantiation into element polyinstantiation.

## ELIMINATION OF ENTITY POLYINSTANTIATION: PART II

| SHIP | | OBJ | | DEST | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Voyager | S | Spying | S | Rigel | S | S |

| CAPTAIN | | RANK | | TC |
|---|---|---|---|---|
| Kirk | U | Admiral | U | U |
| Spock | S | General | S | S |

REFERENCED RELATIONS

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | S | 10 | S | S |

NO ENTITY POLYINSTANTIATION

| CAPTAIN | SHIP | | HOURS/WEEK | | TC |
|---|---|---|---|---|---|
| Kirk | Enterprise | U | 15 | U | U |
| Kirk | Enterprise | U | 10 | S | S |

NO ENTITY POLYINSTANTIATION!

## SUMMARY

- Prevent entity polyinstantiation by proactive (and reactive) mechanisms
- Permit or prevent element polyinstantiation by reactive mechanisms as desired on a case-by-case and day-by-day basis

## DILEMMA

- Design with element-level labels (or more abstract semantic models)
- Emerging platforms provide tuple-level labels

## INTERPRETATION OF TUPLE-LEVEL LABELS

| Starship | Objective | Destination | TC |
|---|---|---|---|
| Enterprise | Exploration | Talos | U |
| Enterprise | Spying | Rigel | S |

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | S | Spying | S | Rigel | S | S |

ENTITY POLYINSTANTIATION

| Starship | | Objective | | Destination | | TC |
|---|---|---|---|---|---|---|
| Enterprise | U | Exploration | U | Talos | U | U |
| Enterprise | U | Spying | S | Rigel | S | S |

ELEMENT POLYINSTANTIATION

## CONCLUSION

- We are optimistic that it will be possible to
  - Design mls databases with element-level labels (or more abstract semantic models)
  - Translate these designs on to emerging platforms with tuple-level labels
- Further R&D is needed to substantiate this claim
- Without such R&D designers of mls DBMSs with tuple-level labels will have a frustrating experience

## Discussion: User Perspective

**Discussion Leader: Jack Wool, ARCA Systems, Inc.**

(paper not available)

Rome Laboratory

Customer Satisfaction Survey

RL-TR-________

Please complete this survey, and mail to RL/IMPS,
26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

---

Organization Name:_____________________________________(Optional)

Organization POC: _____________________________________(Optional)

Address:_______________________________________________________

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful    1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No______

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.

☆U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-20039